
MAGAZINE
MSX
AÑO I
Núm. 3
Julio 1985
250 Ptas.
Casi todo sobre Joysticks
Z80, corazón de león
256 caracteres programables

¡EL IMPERIO CONTRAATACA!

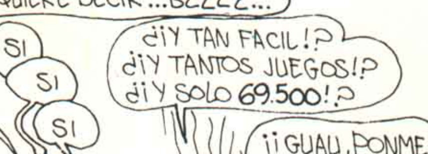












Ordenador Personal
TOSHIBA HX-10
Su Ordenado Servidor

69.500 Ptas.



MSX SYSTEM

Características principales:
Sistema standard MSX. Memoria de 64 K RAM, 32 K ROM y 16 K de pantalla. 16 colores. 73 teclas. 32 sprites. Sistema multicolor: 64 x 48 bloques. Sonido: 8 octavas tres acordes. Conexiones para: casette, impresora, 2 mandos y futuras expansiones.

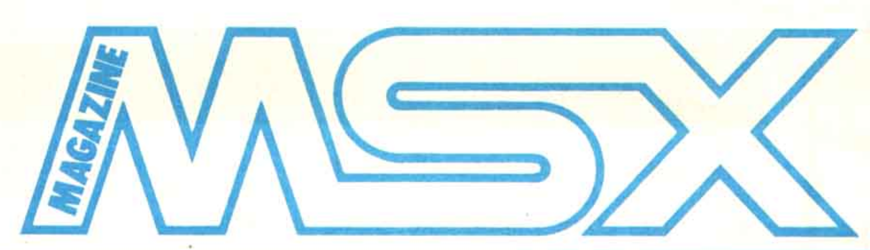


TOSHIBA
española de microordenadores s.a.
Caballero, 79 - Tel. 321 02 12 - Telex 97087 EMOS - 08014 BARCELONA

MSX SYSTEM

El sistema MSX es un standard utilizado universalmente que permite disponer de una gran variedad de programas y accesorios compatibles entre sí.

PUZZLE

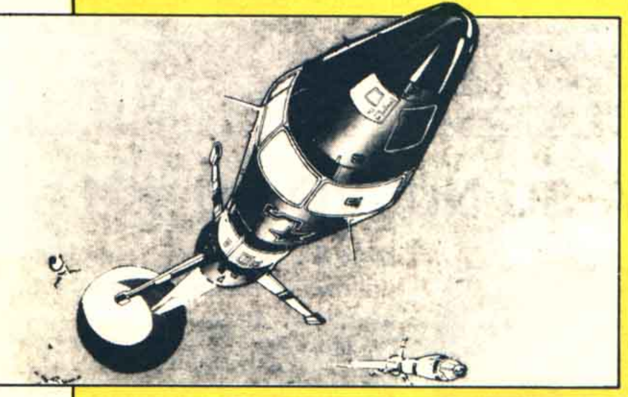


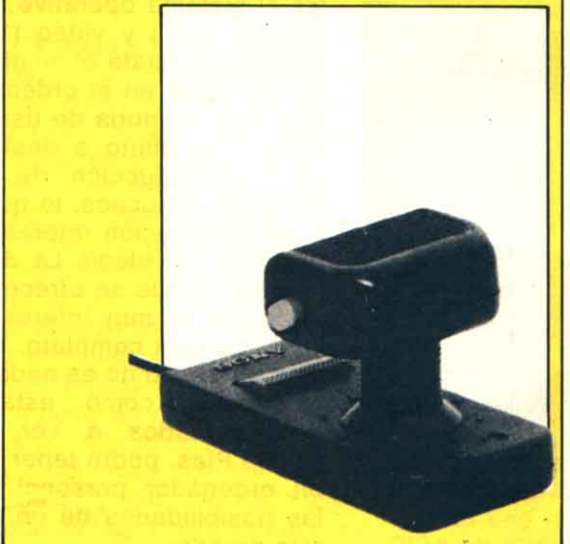
AÑO I - Núm. 3
Julio 1985

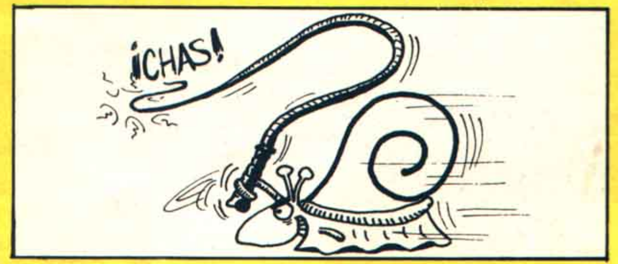
Llegan las vacaciones y con ellas, los ratos de ocio, que mayormente se dedican a la vida placentera y la lectura, y si usted se lleva de vacaciones MSX, lectura no le va a faltar, ya que este mes le ofrecemos temas tan interesantes como el de los joysticks y los caracteres programables. Además como ya anunciábamos, este será un verano caliente en el mercado de los ordenadores, con muchas novedades.

También, habrá tiempo de escribir esos programas que durante el curso no se pudieron realizar, por lo que desde aquí animamos a los lectores y, cómo no, abrimos las puertas de la revista para publicar sus colaboraciones y programas.

¡Hasta el mes que viene!







NOTICIAS

SEGUNDA GENERACION DE MSX

Al igual que los ordenadores de más nivel, el MSX está alcanzando en Japón unas cotas difícilmente igualables.

Efectivamente, Thosiba, ha presentado la segunda generación de ordenadores MSX. El nuevo aparato, HX-22, tiene unas características muy peculiares tales como memoria RAM de 80K compuesta de 64K más 16K de RAM en disco y 64K en ROM, de los que 32K están reservados al BASIC, mientras que el resto de la memoria ROM tiene un procesador de textos. Este procesador de textos, originalmente desarrollado para el IBM ha sido incorporado al ordenador, haciéndole más potente que el modelo anterior.

En otro plano, también se presentó el HX-23, ordenador de la segunda generación con 128K de memoria de video RAM. Este se venderá seguramente el año que viene. Pasado al aspecto de los periféricos, esta marca también presentó, una unidad de diskettes diseñada verticalmente, el HX-101. Este manejará los diskettes de 3.5 pulgadas con un capacidad de 500K sin formatear (360K formateado), de simple cara.

ITAR BAJA LOS PRECIOS

Ya anunciábamos, el número 1 de MSX Magazine, la introducción en el mercado español de un ordenador que, aunque de nombre conocido, no desmerecían sus prestaciones. Ahora, el ordenador que ellos distribuyen, GOLDSTAR FC-200, se vende más barato. Su actual precio, 49.500 Ptas. hará volver la cabeza a más de uno. Con ello se unen a la actual ola de baja de precios iniciada el mes anterior.

SONY LANZA SU NUEVO HB-101



Entre las características principales del ordenador nos encontramos con una memoria ROM de 48K, 32K de ASIC MSX y 16K de programas de utilidad (listín telefónico, agenda y archivo).

Mientras que la memoria RAM tiene 48K repartidas entre el sistema operativo (3K), usuario (29K) y video (16K). Por lo que, hasta el momento se convierte en el ordenador con más memoria de usuario libre. Otro punto a destacar es la introducción de dos ports de cartuchos, lo que le hace una opción interesante a la hora de elegir. La documentación que se ofrece con el equipo es muy interesante y de los más completo. Además su precio no es nada escandoloso, como estamos acostumbrados a ver, por 53.000 Ptas. podrá tener todo un ordenador personal, con las posibilidades de un aparato grande.

DISCOS COMPACTOS, LA ALTERNATIVA DEL DISKETTE

Indudablemente, estos aparatos van a ser el futuro, en cuanto a almacenar información. La intención de los grandes fabricantes es la de crear equipos que puedan trabajar con disco de audio y discos compactos.

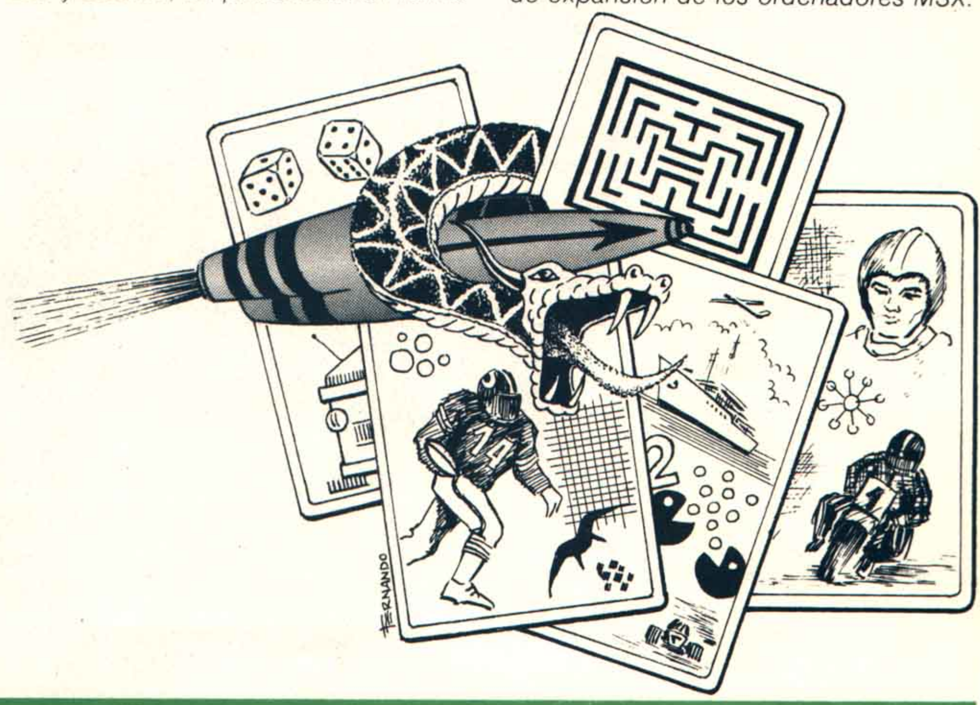
Como alternativa de los diskettes, estos compactos tienen unas garantías que no poseen los anteriores. Algunas características son, su fiabilidad, su solidez y la dificultad existente para piratear programas grabados en estos discos. Pero sin duda alguna, la gran ventaja está en la capacidad, ya que un disco de este tipo puede almacenarse hasta 500 Megabytes.

¿SOFTWARE EN TARJETAS INTELIGENTES?

Lo que nos faltaba por ver ya ha llegado (por lo menos a nuestros oídos). Estábamos acostumbrados a jugar con programas en cinta de cassette y hasta en cartucho ROM, pero esto último ya es el no va más de las innovaciones tecnológicas.

Ahora cualquiera puede camuflar un juego en la cartera, gracias al invento de la compañía Mitsubishi. Esta, después de tres años de continuos esfuerzos y estudio, ha presentado en formato de tarjeta de crédito algunos juegos.

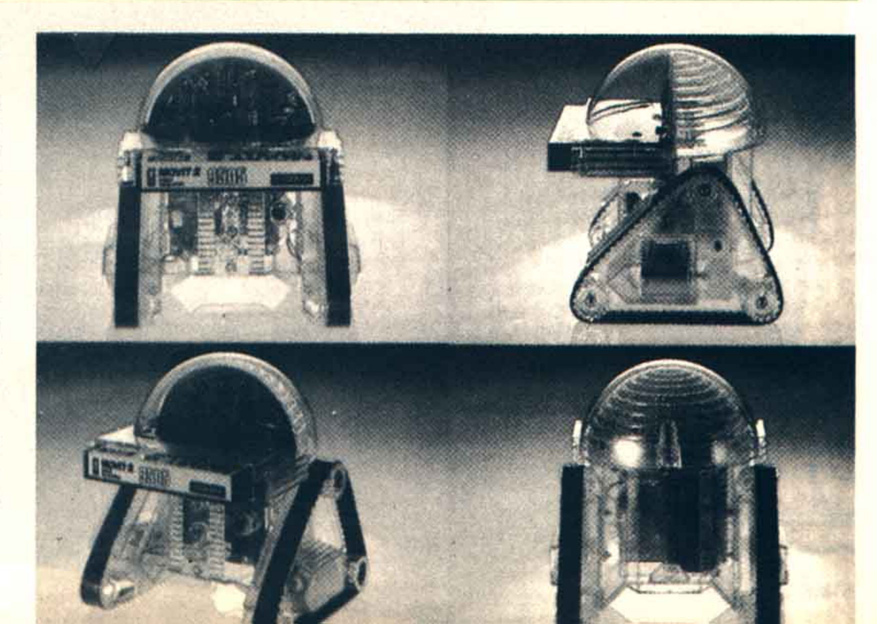
Estas tarjetas tienen unas dimensiones muy reducidas, (1,8 mm capacidad de hasta 1 Megabyte, una memoria ROM programable una sola vez, eléctricamente 256K y otra memoria ROM con borrado eléctrico 256K. Para utilizar esta tarjeta hará falta, como no, un interface especial que se conecta al port de expansión de los ordenadores MSX.



EL ROBOT DEL AÑO 2000, AL ALCANCE DE TODOS

El aparato en cuestión es el Movit 2 9505, que cuenta con diversos aditamentos, como son dos motores para la tracción (uno en el lado izquierdo y otro en el derecho), emite una señal sonora y tres bombillas. La paticularidad de este aparato reside en su programación, ya que ésta se hace con un cartucho ROM preparado a tal efecto, es decir, conectaremos el cartucho al ordenador MSX, haremos un programa cualquiera, con las opciones que nosotros deseemos y lo volveremos a poner en el robot

La posibilidad de utilizar cartuchos para programarlo, permite la realización de programas de 256 líneas con un total de hasta 1024 instrucciones.



JOYSTICKS: COMO SON





Más de una vez nos hemos visto envueltos en auténticas matanzas de seres extraterrestres con nuestro ordenador, e incluso habremos sufrido una muerte vil por no haber sido lo suficientemente rápidos para maniobrar a tiempo y esquivar los furtivos ataques a los que contínuamente nos someten las fuerzas enemigas.

En estos tiempos muchas veces echamos de menos poseer un buen control de la "nave" que estamos pilotando. Para ello, hubo personas que se apiadaron de nosotros y adaptaron artilugios que en un principio no estaban diseñados para jugar, convirtiéndolos en potentes mandos que utilizábamos para controlar nuestra nave sideral. Había nacido el *joystick* y a partir de entonces cambió la suerte de estos extraños seres, que comprobaron en sus **bits** lo peligroso que resulta un humano al mando de un *joystick*.

Bromas aparte, en este artículo intentaremos aclarar los puntos importantes que, relacionados con los mandos externos a teclados, pueden ser algo problemáticos. De esta forma, después de leer estas líneas, puede hacerse una idea clara de lo que

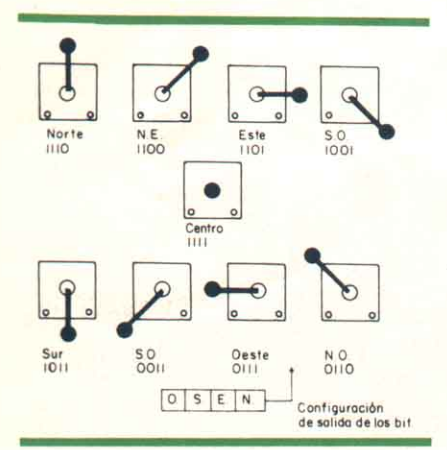


Figura 1. Posiciones del mando de un joystick tipo interruptor.

son, como son y para que sirven, (ya que no todas las aplicaciones son para exterminar bichos raros).

En una primera aproximación, vamos a intentar hablar de ellos en general, para entender su funcionamiento y poderles sacar el máximo rendimiento. Posteriormente, veremos una serie de características de los más variados que hay en el mercado y que están disponibles para que cada uno pueda decidirse por el que mejor le convenga.

Estos *joysticks* los podemos considerar, para comprenderlo mejor, como mandos externos al teclado, que facilitan la comunicación entre el ordenador y usuario (los incrédulos pueden ver aquí una aplicación diferente al típico mando que controla la nave). Son periféricos de entrada que proporcionan al ordenador señales externas que le indican las acciones a tomar.

Como vimos anteriormente, en un principio, no fueron pensados para jugar, sino para ayudar a los expertos que empezaban a trabajar con ordenadores en diseño gráfico. Lo utilizaban de cara a poder posicionar mucho mejor el cursor o modificar ciertas zonas delimitadas por las líneas que formaban la figura. Hoy en día, estas personas usan el lápiz óptico, que presenta ventajas superiores al mando proporcional, que era como al principio se denominó al precursor de lo que actualmente conocemos por *joystick*.

El uso por estos profesionales fue lo que determinó la aparición de los diferentes tipos que hoy conocemos y más adelante pasaremos a comentar. Así, dependiendo del trabajo para que se necesitaban aparecieron distintas modificaciones.

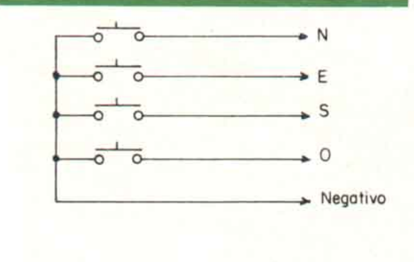
Paddles, trackballs, *ratones y tablas digitalizadoras, nos inundan con sus aplicaciones específicas.*

Hay que tener en cuenta que si los hemos definido como periféricos de entrada, necesitarán unos *interfaces* que adapten las señales emitidas a las que utiliza el ordenador.

Ya dijimos que existen diversos tipos de *joysticks*. Esta variedad se basa en la velocidad de respuesta que posee, es decir, el tiempo que tarda en captar la señal que nosotros comunicamos, transmitirlas al ordenador y estar dispuestos para una nueva señal. Además la diferencia se puede basar en la forma que tienen de captar la señal, aunque, la manera en que esta se decodifica es asunto del *interface*.

Dan la sensación de ser unos periféricos rápidos, ¡craso error! La limitación de velocidad, no es cuestión del aparato, sino de nosotros mismos, de las personas que lo manejan, puesto que en última instancia somos nosotros quienes mandamos la información al ordenador.

Dentro de los *jocticks* podemos defi-



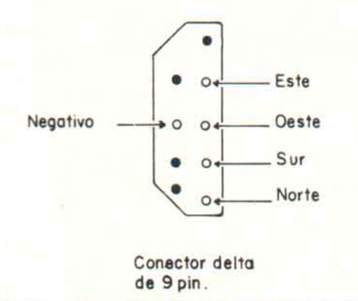


Figura 2. Diagrama esquemático y conexión de los pines de un conector de joystick

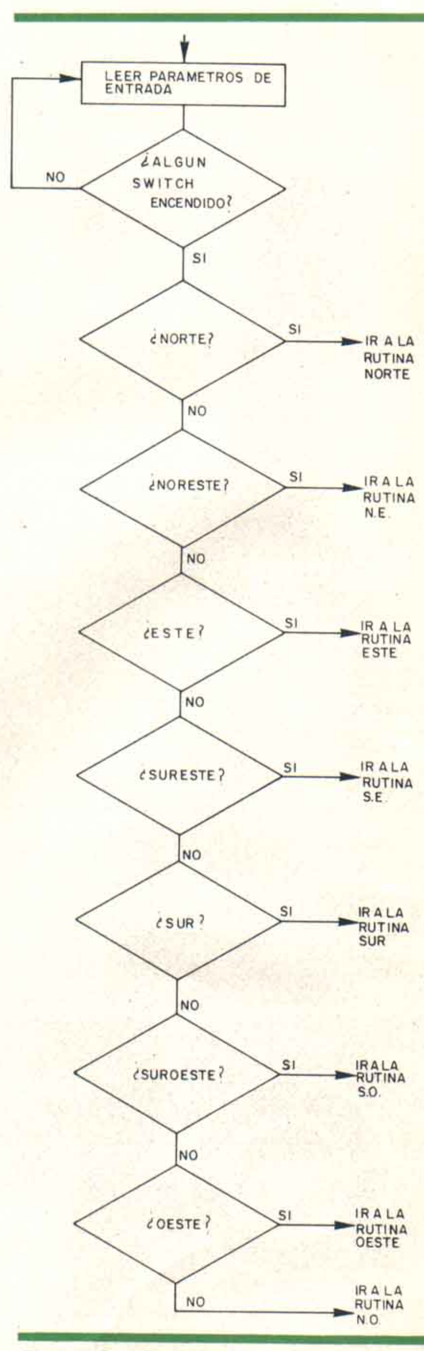


Figura 3. Organigrama de un simple joystick.



nir dos tipos aunque no todos queden enclavados en algunos de estos.
Estos podríamos agruparlos en; .
**TIPO ATARI, ya que esta marca fue la pionera en comercializarlos a gran escala con sus famosos video juegos. Es el típico mando de interruptor.
**TIPO ANALOGICO, algo más complejo que el anterior, se asemeja bastante a los telemando y de hecho son una vesión adaptada a ordenadores, No están tan difundidos como los anteriores. Vamos a tratar en profundidad cada uno de estos tipos,

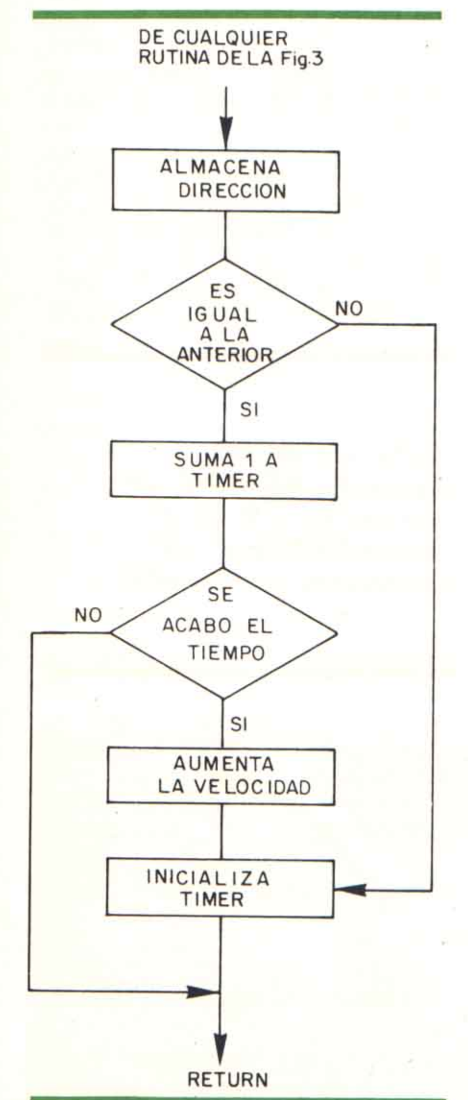


Figura 4. ***Modificación del diagrama de la figura 3, para que informe sobre la velocidad.***

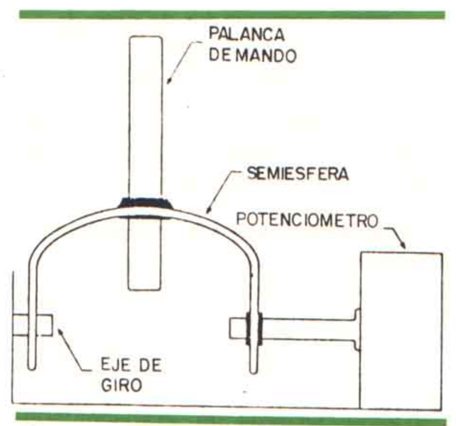


Figura 5. ***Dibujo que muestra la semiesfera de un* joystick *analógico.***

para después hablar de los distintos modelos que podremos encontrar en el mercado, como por ejemplo los *paddles*, bolas *(trackball)*, ratones y placas.

Joysticks del tipo Atari

Los *joysticks* del primer tipo, hemos dicho que son los más extendidos, de hecho en el mercado prácticamente son los más asequibles, ya que los otros, de existir suelen tener

Son los periféricos más populares, por excelencia, en el mercado del ordenador personal.

precios prohibitivos dada su poca comercialización y difusión.
Si alguien se le habla de un *joystick*, por poco que haya oido o jugado, se imaginará inmediatamente un mando de este tipo lo que indica su extensión. Aunque existen muchas variedades, como veremos en futuros artículos, todos ellos constan de las mismas partes, pudiéndolo generalizar en un tipo común.
Estos son de tipo interruptor porque cada movimiento del mando actúa y modifica una serie de interruptores internos, trasmitiendo al ordenador, via *interface*, las coordenadas correspondientes.

La mayoría de los lectores ya sabrán como es un *joystick* y mas de uno sabría manerjarlo al momento. Su funcionamiento es sencillo y las direcciones en que podemos dirigir nuestra "nave" o el mismo cursor las podemos ver en la figura 1. También podemos observar el valor, en binario, de cada dirección posible. Este valor es el que hace imposible la compatibilidad de algunos *joysticks* con diversos juegos. De tener todos los mandos los mismos valores, podríamos conectarlos a cualquier juego. Lo único que sí tomaron en cuenta los fabricantes de estos periféricos, fue el conector de 9 patas (figura2) que apareció en el **Atari** 2600VCS. Sin embargo, no todos han conseguido esta pauta y han decidido comercializar "su" *joystick*.
En diversos modelos, el *interface* que conecta el mando con el ordenador es simple. Si consideramos las cuatro direcciones cardinales, existirán 16 combinaciones posibles, que con un conmutador de nueve patas es posible aprovechar, así, con cualquier ordenador que posea un *port* paralelo de cualquier tipo, podrá utilizar este mando sin "adaptador" o *interface*. Si el ordenador no dispone de este *port*, el *interface* deberá ser el que posea dicho conector.
En el esquema de la conexión, destacaremos dos líneas diferentes según el valor del voltaje que posea;
— Una línea común, que generalmente es negativa e indica al ordeador que no se ha indicado orden alguna. Es decir, sirve

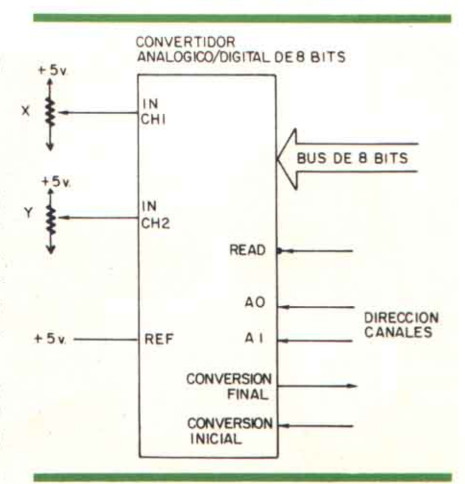


Figura 6. ***Diagrama de la conexión típica de un* joystick *de 2-D.***

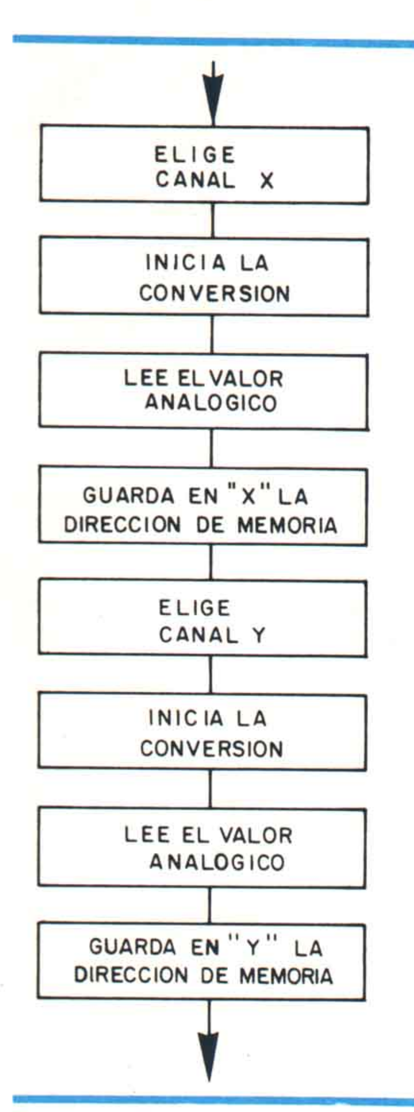


Figura 7. Organigrama de un* joystick *analógico.

como comparador de entrada.
— Una línea con voltaje de signo opuesto que indica orden de movimiento.

Con un sistema de este tipo, el *software* que se necesita es tan simple como el *hardware* utilizado, así, con la rutina siguiente, es posible leer las instrucciones que, vía *joystick* introducimos;

IN (num. del port), A: Almacena en A la entrada del *port* n,

CP x: Comprueba si la orden es x,

JR Z x1: Salta a la rutina que desplaza a x,

. . . .

RET

Este sería un ejemplo en Assenbler; en BASIC aunque más fácil, resulta más lento. El diagrama de flujo de esta rutina lo podemos ver en la figura 3.

Continuando con los distintos *joysticks*, llegamos a uno muy peculiar, cuya característica más importante era el aumento de la velocidad del objeto mientras se mantenía pulsada una dirección concreta. Es del mismo tipo que el anterior, lo único que varía es el *sofware*, que por medio de ciclos, aumenta la velocidad cada vez que se repite la misma orden. Es decir, si en dos ciclos consecutivos, el ordenador lee la misma dirección, éste aumentará la velocidad del objeto. Un esquema claro de como funcionan lo podemos ver en la figura 4. Con este sistema se crea un efecto analógico con un aparato digital.

Puede ser interesante dar una idea de la diferencia que hay entre un dispositivo analógico y otro digital, no dando la típica descripción sino desde la visión de un profano entienda. Así, un dispositivo analógico presenta un margen infinito en la escala que mide voltajes o intensidades, mientras que uno digital solo mide múltiplos enteros de una unidad de partida. Vamos a poner un ejemplo que aclare este concepto. Imaginemos un aparato que mida voltajes, como puede ser un voltímetro o VU-metro de los que vienen en equipos de alta fidelidad. Hoy día es posible disponer o aún cuando menos, haberlos visto alguna vez. Los analógicos tienen una aguja que mide sobre una escala y los digitales llevan un cierto número de *displays* numéricos. Al ir variando el voltaje, es posible observar que la aguja se mueve lentamente, mientras que el otro caso no varía, es decir, por muchos displays que dispongamos, siempre el dispositivo analógico apreciará diferecia en el voltaje, mientras que el digital puede quedar dentro de un error o rango de medida. Un ejemplo, disponemos de un voltímetro de aguja y uno digital que es capaz de medir hasta 0.1 volt. Si la variación del voltaje es 0.01, la aguja se desplazará levemente, mientras que en el digital seguirá igual ya que para el 0.00 y el 0.01 el voltaje es el mismo, siempre y cuanto esté bien calibrado.

Partiendo de los diseños CAD/CAM, se han convertido en el "arma mortífera" de los juegos de invasión.

Joystick del tipo analógico

Es algo más elaborado desde el punto de vista del mando, además su *interface* no es tan simple como en el caso anterior. Esta es la razón de la mayor difusión del tipo interruptor



Módulos de Joysticks



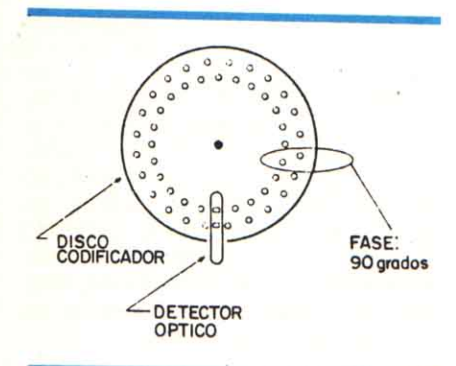


***Figura 8. Configuración básica de un* joystick *tipo* trackball.**

frente al analógico. Normalmente, usa dos o más resistencias conectadas a sistemas mecánicos de movimiento que simulan las direcciones. Este sistema se denomina, aros o cercos y delimitan el posible movimiento. Al mover el *stick* de control se mueve uno o varios de estos cercos, de forma que también se mueven los potenciómetros o resistores. Para los que se inician en la materia, un potenciómetro o resistor, es una resistencia variable, analógica, a la que estamos acostumbrados ya que todos los controles de volumen que poseemos en los electrodomésticos son así. Un esquema general de este *joystick* lo podemos ver en la figura 5.

Hemos resaltado que los potenciómetros son sistemas analógicos, por lo que aquí surge la diferencia con el tipo anterior, mientras que en el otro, a cada movimiento se pulsaba un interruptor, es decir, se mandaba o no se mandaba una señal, aquí se está mandando la señal contínuamente, lo único que varía es la amplitud. El ordenador no entiende todo el rango de voltajes, sino sólo una gama de señales que van, por un lado desde 0 a 3.5 y por otra parte, valores por encima de 3.5.

Ocasionaríamos un desastre si directamente conectamos un mando de este tipo a un ordenador, ya que éste no iba a entender la mayoría de las instrucciones. Sólo comprendería aquellos valores extremos, lo cual no sería rentable, puesto que para ello adaptaríamos el sistema anterior, dado su menor coste. Se hace imprescindible el disponer de un dispositivo convertidor analógico/digital, que transforme el margen que nos proporciona en una serie finita de instrucciones que el ordenador sea capaz de comprender. Este convertidor sería el *interface*. Muchos de estos convertidores se pueden encontrar en circuitos integrados simples que funcionan a 5 v. que es un voltaje común en microordenadores. Hemos dicho que en un sistema analógico, las posibilidades de cambio son infinitas. Si ahora introducimos un sistema que va a pasar de analógico a digital, perderíamos esa ventaja, llegando a un resolución finita. Pasaríamos a poseer un recorrido finito de posiciones de mando, en lugar del valores infinitos como sería deseable. No obstante estos convertidores suelen dar una buena resolución y por ejemplo, mandos de este tipo que se utilizan en diseños gráficos suelen tener 256 posibilidades. Queda claro entonces que las posibilidades de este sistema presenta mayor

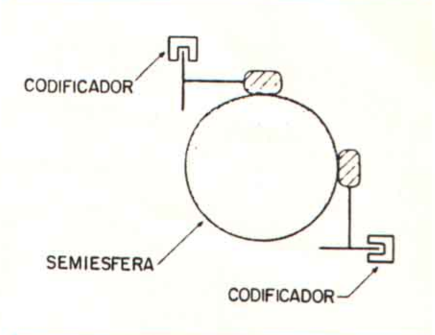


Figura 9. Aumento del disco que codifica la dirección, utilizado en* joystick *de la figura 8.

potencia a la hora de utilizarlo, sin embargo hay que sopesar esta ventaja con el *sofware* necesario para interpretar la señal que proviene del convertidor.

En estos sistemas, cuantos más potenciómetros se añadan, de más dimensiones se dispondrá , aunque el interface, lógicamente, se complicará con la presencia de un nuevo convertidor. Esta tercera dimensión se puede usar como efectos zoom, mover objetos, etc.

Existen diversos tipos, analógicos y digitales, que sirven para situar el cursor en cualquier lugar de la pantalla.

En el diseño del *sofware* que interpreta la salida de este mando, se debe introducir la comparación de los valores que se leen en este momento, con los valores obtenidos anteriormente. En este caso, la dirección inicial al empezar a utilizar el mando es totalmente aleatorio. Si el valor que se mide es mayor que la dirección en que se mueve, entonces el movimiento es positivo, considerando coordenadas cartesianas y los potenciómetro definiendo las cooordenadas del sistema, con la escala ya definida por el *interface*.



Así queda el Joysticks tras una jornada de "trabajo"

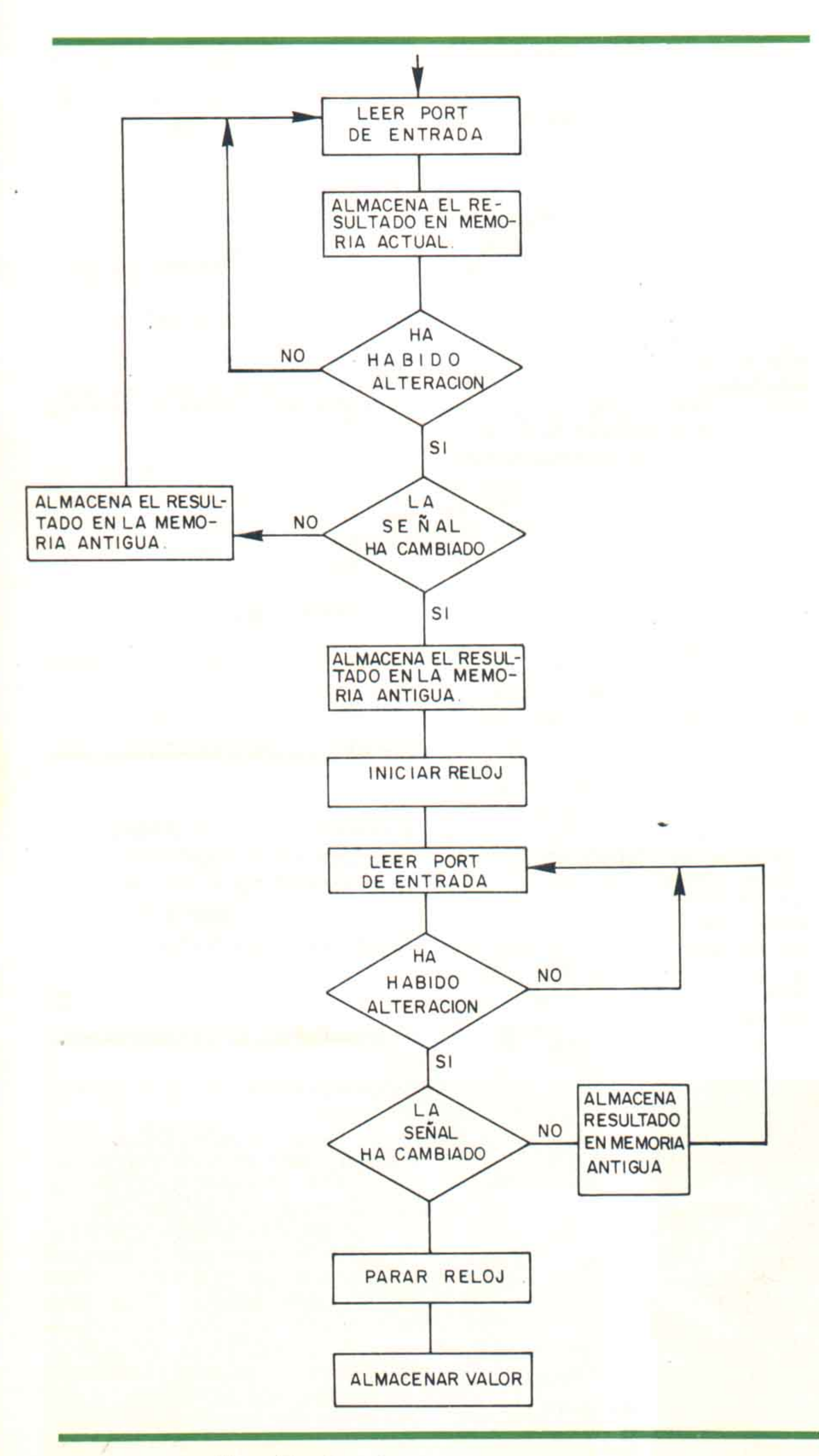


***Figura 10. Organigrama del* joystick *tipo* trackball.**

Se podrá notar, que en ninguno de estos tipos se ha nombrado algo a lo que estamos acostumbrados, los disparos, ya que por muchos movimiento que imprimamos a la nave, si no disparamos es como si no hiciéramos nada. Esta especie de gatillo no es más que un simple pulsador, en uno y otro sistema, por lo que todo lo explicado en el primer apartado, también sirve para este. El disponer de uno o más disparadores no es más que apurar el diseño.

Otros tipos de joysticks

Una vez introducido someramente el funcionamiento general de estos mandos, vamos a detallar algunos de los más conocidos, destacando sobre todo propiedades particulares que no hallamos especificado anteriormente.

Paddles

Fueron los primeros utilizados. Son analógicos, con dos direcciones y cuatro sentidos, arriba, abajo, derecha e izquierda. Podemos distinguir entonces dos tipos, unos los más generales, que responden a esta descripción y otro mando, pero con un volante que se mueven 360 grados como los potenciómetros.

Normalmente se necesitan complicados sistemas de conversión analógico digital para cambiar valores numéricos con posibilidad de aumentar o disminuir la magnitud medida. Cuando desee realizar un efecto como la rotación de un objeto, tendrá que echar mano de *sofware* más complicado. Hoy día, el uso de este tipo de mando está muy restringido en ordendores personales, quedando relegada su función a máquinas comerciales que vemos en las salas de juegos. En realidad, lo más utilizados hoy en día pueden parecer versiones modernas de aquellos, pero en realidad, responden más al tipo de **Atari** ya explicado ateriormente, aunque en apariencia parezca lo mismo, mientras que el *interface*, sí es fácil diferenciarlos, dada la mayor complicación del que posee un *paddle* frente al otro.



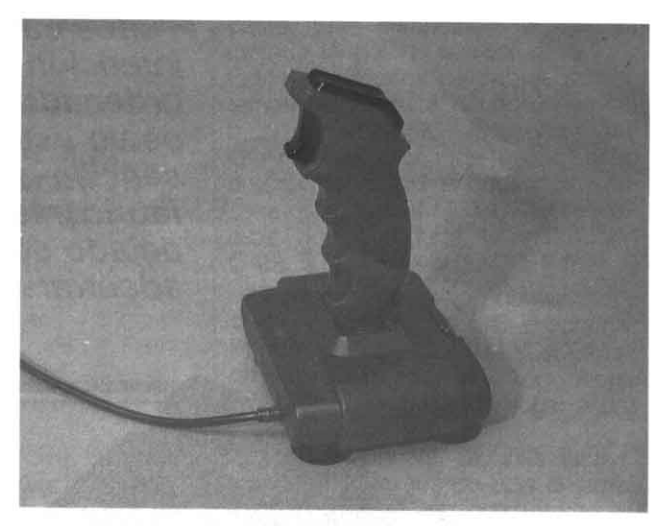
Trackball

Se les conoce como bolas. Aparecieron hace algún tiempo en las máquinas de juegos que veíamos en los bares.
Curiosamente, esta bola daba la misma información que un *joystick*. De hecho, esta información, se enviaba con más rapidez. Es un dispositivo cuya disposición podemos ver en la figura 8. Externamente, se parece a una bola de billar, que al rodar dentro del soporte, acciona dos potenciómetros conectados a su vez a sendos decodificadores ópticos. Las señales enviadas por los decodificadores se mandan a un circuito integrado, donde se convierte la señal equivalente enviada por cualquier *joystick* del tipo interruptor.
Un programa para controlar este tipo de *joystick* lo podemos ver en la figura 10. Tendremos en cuenta que cuando la señal mandaba hacia un sentido era pequeña, es que estamos cambiando la dirección. El *software* suele ser complicado, aunque el programa se puede hacer a medida. Expliquemos el diagrama de la figura. La primera acción que se realiza es la lectura del estado de cada señal de entrada y almacenarla en la memoria. Esta señal, en función de comparaciones sucesivas va a determinar la dirección a tomar. A partir de esto, cualquier señal recibida a continuación es comparada con la existente para la tomar la decisión de continuar en esa dirección o alterarla.
Al recibir la señal, tendremos en cuenta que se van comprobando una serie de parámetros, como son la velocidad de la bola, el sentido, etc.

Ratón

A pesar de su nombre tan extraño, cualquier usuario de este tipo de *joystick* estará muy contento de poseerlo. Util y cómodo de utilizar, con diversos modelos, podemos decir que su funcionamiento es similar al modelo anterior.

En este caso, la bola está invertida con una serie de pulsadores, uno o dos, en la parte superior. Al ser similar al anterior, es comprensible que éste posea un convertidor analógico/digital.
De cualquier manera, este mando no es muy corriente ni está muy generalizado, además su finalidad no está en los juegos, sino en otras aplicaciones en las que no vamos a entrar.



Quickshot II

Tablas digitalizadoras

Desde tiempos inmemorables, el hombre desarrolló la habilidad de escribir. El papel y lápiz han estado siempre a mano, por lo menos hasta que llegaron los ordenadores. Entonces se dejaron a un lado las calculadoras, máquinas de escribir y ahora, gracias a este invento, están a punto de dejar de lado las mesas de diseño. Hay tamaños diversos y muy variados, pero todos con el mismo principio.
Se trata de un tablero de reducidas dimensiones, con varias capas de peliculas resistentes. Apretando el lápiz que viene, contra el tablero veremos como en la pantalla aparece el cursor, como si de un *joystick* normal se tratara.
Señales ultrasónicas emitidas por unos muelles dentro de la tabla, son recogidas por el sensor situado dentro del "lapiz" y según la fase en la que se encuentra la señal, el ordenador podrá determinar la situación del cursor.
El *interface* de un tabla de este tipo es más complicada que en todos los casos anteriores, solo el conector ya es un mundo, de manera que introducirse en las cualidades técnicas sería complicar el asunto.



Joysticks fabricado por SONY

Todos sabemos que el MSX, se concebió con la intención de unificar los criterios al diseñar ordenadores personales. Pero ¿hasta qué punto es un estándar? La respuesta no parece sencilla, sobre todo si la aplicamos a los fabricantes de diskettes, puesto que ellos han dejado de lado un poco la estandarización, para adentrarse más en sus intereses.



Los diskettes de 3.5 pulgadas son, hasta el momento la mejor opción.

MSX-DOS ¿diskette standar?

Aunque el estándar, lo es en cuanto al sistema operativo, no lo es tanto en lo que se refiere al formato de los discos. Basta acercarse a cualquier concesionario de ordenadores para ver que cada ordenador propone su formato de disco como el mejor. Así podemos ver *diskettes* de múltiples tamaños, desde el "enano" de 2.8 pulgadas de **Mitsumi,** hasta el normal de 8 pulgadas, pasando por los de 3 y 5 pulgadas sin olvidar el de 3.5. Este último, parece ser que se va a imponer. La razón hay que buscarla en otro tipo de ordenadores, la nueva generación de los IBM PC. En efecto, parece ser que la multinacional americana va a lanzar sus nuevos PC con *diskettes* de este formato, lo que redundaría, indudablemente, en la comercialización de los 3.5 pulgadas y más en los *disc drives* de **Sony,** que ya llevan tiempo en el mercado, con la correspondiente experiencia. Cuando esto ocurra, podrá introducir el *diskette* de datos, grabados en el **IBM,** a un ordenador MSX y leer los datos con el programa adecuado.

Abandonemos este tema por el momento.

Los que hayan trabajado con ordenadores de más nivel, sabrán por experiencia que el DOS es uno de los mejores sistemas operativos de discos que hay, de hecho, es el más utilizado por los profesionales. No cabe la menor duda, de que teniendo un pasado tan interesante y un futuro prometedor, debido a las constantes mejoras a que es sometido, como lo ha tenido el MS-DOS, sea este el sistema operativo adoptado por el estándar MSX **(Microsoft,** también tiene mucho que ver con la elección).

No sorprenderá a nadie que sepa que MS significa **Microsoft,** saber que MSX-DOS, es muy similar en la forma en que trabaja.

El ordenador detecta la conexión del *interface* a la expansión del cartucho del ordenador y pone en funcionamiento el BASIC de disco. Existen dos clases de BASIC a ejecutar con su propio juego de instrucciones: MSX-DOS y BASIC MSX de disco. Para ejecutar el segundo modo, hay que introducir directamente el comando BASIC y pulsar RETURN a continuación.

La variedad diskettes en el mercado, único punto negativo de la compatibilidad absoluta.

BASIC MSX de disco

Primero veamos los programas creados con el BASIC MSX de disco ya que es con el que primero se trabaja. Se utiliza a diario en la labor de programación, especialmente cuando el soporte externo no permite realizar filigranas, el cassette por ejemplo. Los programas especialmente dedicados a aplicaciones concretas se han de grabar con instrucciones como AUTOEXEC.BAS, de esta forma, al conectar el ordenador, este programa se ejecutará directamente, ya sea un tratamiento de textos o una base de datos. Esta instrucción se utilizará cuando tengamos un *disc drive.*

Estos dos modos de trabajo son alternativos, es decir, si uno desea trabajar con el BASIC MSX tendrá que introducir el comando especificado anteriormente, por el contrario, si deseamos volver al MSX-DOS, tendremos que introducir la instrucción CALL SYSTEM y pulsar RETURN. Si ejecuta esta operación con un disco que no contenga los programas MSXDOS.SYS y COMMAND.COM se producirá un error del tipo *"Illegal Function Call"* (llamada ilegal a una función).

Empecemos a ver algunas instrucciones que el mes pasado dejamos en puertas, para ser comentadas en este número. Vamos a iniciar el tema comentando la instrucción que formatea los discos. Esta es esencial ya que prepara los *diskettes* para su utilización. Para formatear un disco desde el BASIC MSX de disco o el MSX-DOS, usaremos la instrucción FORMAT. Esta es algo distinta según el modo en que estemos trabajando. Si lo realizamos con el primer modo, introduciremos la instrucción CALL FORMAT, mientras con el MSX-DOS nos bastará con teclear el comando FORMAT directamente. Tener en cuenta que el *diskette* se borrará completamente al efectuar un FORMAT.

Pasemos a ver las instrucciones de



Unidad de diskette de Sony de 3.5 pulgadas e interface.

Entrada/Salida de datos, SAVE y LOAD.
Algunos de los comandos usados del BASIC de disco es muy similar al utilizado en el BASIC de *cassette*. Instrucciones como SAVE, LOAD, BSAVE, BLOAD, INPUT y PRINT, funcionan de la misma manera que en *cassette*, con la excepción de que el prefijo "CAS:" se sustituirá por la letra "A:" o "B:" según el *diskette* que se está utilizando y si hay dos unidades de discos.
En este BASIC, la instrucción SAVE sería de la siguiente forma:

SAVE "XX: NOMBRE. TIPO"

donde XX indica el nombre del dispositivo (A o B). NOMBRE es el del fichero a grabar y no puede tener más de 8 caracteres, por último, TIPO es una descripción del tipo de programa que se trata, ya sean datos, ficheros o programas, en código máquina o en BASIC. Esta descripción se puede omitir, pero es muy útil a la hora de encadenar programas con los datos adecuados. Por ejemplo, si tenemos una pequeña agenda telefónica y por otro lado, tenemos la dirección, podríamos tener el programa grabado bajo el nombre de AGENDA.BAS, y los datos de las direcciones se podrían grabar con el nombre de AGENDA.DAT.
Si le da un nombre superior a los ocho caracteres permitidos, el ordenador insertará un punto detrás del octavo caracter y asumirá los tres siguientes como identificador del tipo de fichero y si hay más de 11 caracteres, a partir del 12 en adelante se ignorarán.
Se pueden utilizar letras, mayúsculas o minúsculas y números para darle nombre a los programas, apareciendo todos en mayúsculas al pedir el directorio del disco.
La sintaxis de la sentencia LOAD, es similar a la de SAVE.
La siguiente instrucción a tener en cuenta es la que ayuda a fusionar ficheros con datos o ficheros entre sí, esta es la sentencia MERGE.
Este comando permite añadir a un programa ya existente en la memoria, otro programa, siempre y cuando se trate de ficheros de programas en formato ASCII, que se hayan grabados con la instrucción;

SAVE "XX: NOMBRE. TIPO", A

En este caso, la mejor solución es colocar como identificador las letras ".ASC", para evitar, que por error, se fusione con un programa en BASIC.
Es posible fusionar programas con los mismos números de sentencia pero para evitar el confusionismo que esto conlleva, antes de realizar esta operación, hay que reenumerar con el comando que existe expresamente para ello, RENUM.
Además, es posible ejecutar los programas según se cargan, sin necesidad de teclear la instrucción. Para ello en lugar de cargarlo con el comando LOAD, se hace con el comando RUN, obteniendo así el mismo resultado. Aunque, si grabamos un programa, con la letra ".R" al final, veremos cómo se ejecuta igualmente. Este caso lo habrá visto alguna vez, ya que es la forma en que cargan algunas cintas comerciales.

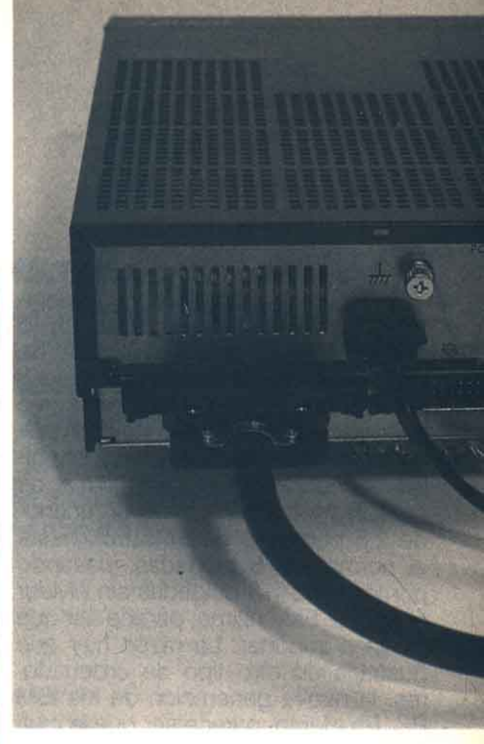
El pequeño ordenador con un gran sistema operativo, que está constantemente renovándose.

Tratamiento de ficheros

El directorio de un *diskette* se obtiene con el comando FILES.
Este mostrará el directorio completo del diskette elegido, por el contrario, para borrar un fichero que no nos interese, introduciremos el comando KILL. Su sintaxis es algo más escueta que para cargar o para grabar.

KILL "XX: NOMBRE. TIPO"

Este comando permitirá crear espacio en el *diskette*, cuando éste se encuentre al límite de su capacidad.

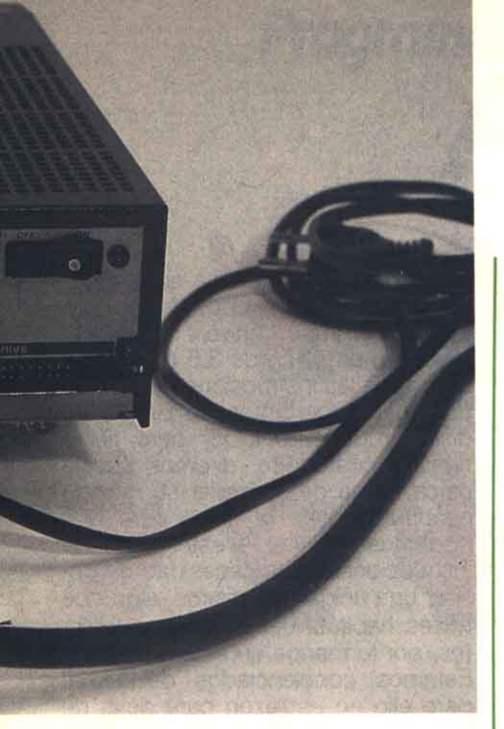


Vista posterior de la unidad de diskettes, con c

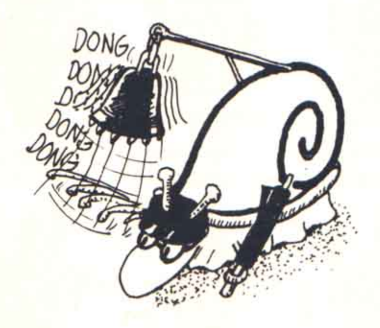
A veces nos hemos encontrado con la necesidad de alterar el nombre de un programa y aunque era una labor algo pesada, teníamos que realizar una serie de operaciones para llevar a cabo tal cambio. Estas operaciones eran las de cargar el fichero (o programa), grabarlo con el nombre nuevo y borrar el que poseíamos con el nombre antiguo. Como vemos, son tres pasos a seguir, algo pesados. Sin embargo, existe una instrucción que permite resumir estos tres pasos a uno solo. NAME, cuya sintáxis es la siguiente, es la que realiza el cambio de nombre.

NAME "NOMBRE ANTIGUO. TIPO" AS "NOMBRE NUEVO. TIPO"

Ahora veamos una instrucción que a más de uno le resultará familiar, el comando COPY. Esta realiza una copia de un programa o fichero directamente. Es a su vez, un comando muy flexible, puesto que permite copiar programas o datos, además según la opción utilizada dentro de esta instrucción, podremos copiar uno o todos los ficheros, en el mismo o en otro *diskette*, en código máquina o en BASIC, resumiendo, con este comando, agilizará la duplicación y las copias de seguridad de los datos y programas más importantes.
Por ejemplo, si deseamos duplicar un fichero en el mismo disco, haremos la siguiente operación,



nexión adicional para otra unidad.



COPY "A: *.BAS" TO B:

copiará todos los ficheros escritos en BASIC de un *diskette* a otro.

Al igual que con los *cassettes,* los comandos BLOAD y BSAVE, se pueden utilizar para los programas en código máquina o para los datos simplemente. El añadir la "R" al final de la instrucción BLOAD, posibilitará la autoejecución del programa después de cargarlo y, si en lugar de ésta, pusiéramos una "S", entonces cargaríamos datos a la memoria RAM de video.

Sin embargo, el comando BSAVE, requiere, especificar las direcciones de memoria para grabar la información, estas son el límite inferior y el superior de memoria que se van a grabar con una dirección de arranque. En caso de omitir la dirección de autoejecución, el ordenador asumirá el límite superior como tal.

FICHEROS DE DATOS

El resto de las instrucciones del BASIC de disco, hace referencia a los ficheros, ya sean éstos, secuenciales o aleatorios.

La diferencia existente entre ellos está en que en el primer caso, si introducimos los datos por orden alfabético, al querer acceder a cualquier registro cualquiera, por ejemplo, uno que empiece por Z, tendremos que recorrernos todo el fichero, desde la A hasta la Z para consultarlo. Mientras que en el segundo caso hay una para acceder directamente al registro deseado. Pero, como todas las cosas, estas dos organizaciones de ficheros tienen sus pros y sus contras.

Los ficheros secuenciales son más simples de crear y más fáciles de entender, mientras que los ficheros aleatorios, son complicados de entender y más aún de crear, en el primer caso es lento en el acceso a un registro, el segundo es más rápido, etc.

Ambos tipos de ficheros utilizan las mismas instrucciones para abrir y cerrar ficheros, OPEN y CLOSE. También, se puede acceder a un número determinado de ficheros a la vez. Esto es posible gracias a la instrucción MAXFILES, donde indicaremos al ordenador el número de ficheros a tratar. Gracias a ella, el ordenador reservará un espacio determinado en la memoria que se manejará como un *buffer.* Hay que tener en cuenta, que cada fichero abierto (OPEN) tendrá un número asignado, que no se podrá utilizar en otro fichero mientras esté abierto. Lo que sí se puede hacer, es cerrar uno y utilizar el número que poseía para abrir otro.

Un completo juego de instrucciones que ayudará a la creación y consulta de ficheros, ya sean secuenciales o aleatorios.

Volviendo al tema de los *buffer,* estos están capacitados para mantener un total de 255 *bytes,* que una vez completos se volcarán al *diskette* o en su defecto, cuando se cierra el fichero. Sin embargo, el usuario tendrá la impresión de que el fichero se ha escrito inmediatamente.

Como es lógico, tiene que existir una forma de indicar que los ficheros secuenciales se abren para leer (entrada de información) o para escribir (salida de información). Esta forma se denomina modo de acceso y su formato es;

OPEN "XX: NOMBRE. TIPO" FOR MODE AS NUMERO-DE-FICHERO

Los datos escritos con la instrucción PRINT NUMERO-DE- FICHERO, VARIABLE, usando el mismo número que el ya existente cuando se especificó el fichero en la instrucción OPEN. Veamos un ejemplo,

PRINT 1,D$

imprime una cadena de caracteres (variable alfanumérica) al fichero. Una vez se han introducido todos los datos necesarios, se ha de cerrar el fichero con la instrucción CLOSE, de la manera siguiente;

CLOSE 1

siendo el 1 el número de fichero que se dio en la instrucción de apertura de ficheros (OPEN).

```
10 OPEN"A;Agenda.Datos" FOR OUTPUT AS#1
20 INPUT "Dato";A$
30 IF A$="XXX" THEN 60
40 PRINT 1,A$
50 GOTO 20
60 CLOSE#1
70 END
```

Esta sería la típica rutina de salida de datos. El comando PRINT 1, D$, es el que se encarga de realizar la operación de grabar. La rutina que lee los datos grabados por el programa anterior es similar a ella.



Ahora que sabemos las instrucciones necesarias para abrir y cerrar un fichero y cómo escribir en él, veamos la manera de añadir registros a un fichero ya existente. Para ello tendremos que echar mano del comando APPEND, que lo pondremos en su lugar correspondiente, con la instrucción OPEN, de la siguiente forma;

```
OPEN "XX: NOMBRE. TIPO"
FOR APPEND AS NUMERO-
DE-FICHERO
```

El resto de un programa creado con esta instrucción es igual que en el primer caso.
Ahora bien, con los ficheros aleatorios, no tienen porque definir su modo, pero sí han de definir la longitud de los registros, esto se hace con la instrucción ",LEN = n" al final del comando OPEN.
Si no se define la longitud de un registro, el BASIC de disco asumirá el valor de 255 *bytes* por defecto.
Para crear un fichero con una longitud de 70 caracteres por registro, haremos lo siguiente,

```
OPEN "XX: NOMBRE. TIPO"
AS 1, LEN = 70
```

Para escribir datos en un fichero, el BASIC de disco tiene un comando que efectúa esta operación, FIELD define la longitud de la variable en el registro. Ahora bien, no se puede escribir directamente en el registro. Hay que enviar la información previamente al *buffer* de salida (memoria intermedia) y a continuación volcarlo o escribirlo en el registro. Estos dos últimos pasos se hacen con las instrucciones LSET y PUT. Esta es la forma a seguir por el programador para crearse un fichero y escribir en él. Para leer los datos de un fichero ya creado utilizaremos la instrucción GET, cuya sintaxis es la siguiente.

```
GET NUMERO-FICHERO, NU-
MERO-REGISTRO
```

El número de registro más alto se calcula dividiendo la función LOF entre el número de *bytes* por registro. LOF son las iniciales de Length Of File (Longitud del fichero).
El comando anterior FIELD, sólo puede manejar variables alfanuméricas, las variables numéricas se tienen que convertir con la instrucción LSET y RSET usando las variables reservadas MKI$ (para variables enteras), MKS$ (para variable en simple precisión) y MKD$ (para variables en doble precisión). Estos dos comandos permiten almacenar la información, con una característica muy particular y es que según la instrucción utilizada, los datos se almacenan alineados a la izquierda o derecha respectivamente.

Los ficheros de datos, programas y archivos, se manejan como si de un "mainframe" se tratara.

El significado de variables en simple y doble precisión lo veremos más adelante.
Por último, nos queda ver una instrucción no tan importante como las anteriores, pero fundamental para el buen hacer del BASIC de disco y el MSX-DOS. El comando DSKF, nos da el espacio remanente en el disco en unidades de grupo, donde una unidad de grupo equivale a dos sectores (en un disco de 3.5 pulgadas).
Estas son las instrucciones y comandos más importantes de BASIC de disco y del MSX-DOS, pero fiel a nuestra costumbre, diremos que la única forma de aprender su manejo es utilizándolos. El único pero que podemos achacar a este juego de instrucciones es la necesidad de poseer una unidad de discos, algo que no es habitual entre nuestros lectores, por lo menos en este momento. Estamos concienciados del tema, pero ello no es razón para dejar de estudiar y comprender las interesantes instrucciones que hemos visto. Además, el MSX es un sistema con unas posibilidades muy completas y por supuesto, el más potente de su categoría.
Conocerlo ayudará a los lectores a indagar profundamente en las posibidades de este aparato.



Programas Sony para ordenadores MSX

A la orden.


Monkey Academy


Países del Mundo-1


Países del Mundo-2


Computador Adivino


Computer Billiards


The Snowman


Cubit


Character Collection


Stop the express (Para el Tren)


Hustler (Billar Americano)


Data cartridge


Quinielas y Reducciones


Home Writer


Sparkie


Aprendiendo Inglés-1


Binary Land


Creative Greetings


Aprendiendo Inglés-2


Antartic Adventure


Mastermind


Contabilidad Personal


Athletic Land


E.I.


Ficheros


El Ahorcado


Dorodon


La Pulga


Cosmos


Control de Stocks


Battle Cross


Mouser


Crazy Train


Ali baba


Juno First


Car Jamboree


Tutor


Track and Field-1 (olimpiadas)


Blackjack


Track and Field-2 (olimpiadas)


Driller Tanks (Tanque Destructor)


Sonygraph


Ninja (El Samurai)



Les Flics



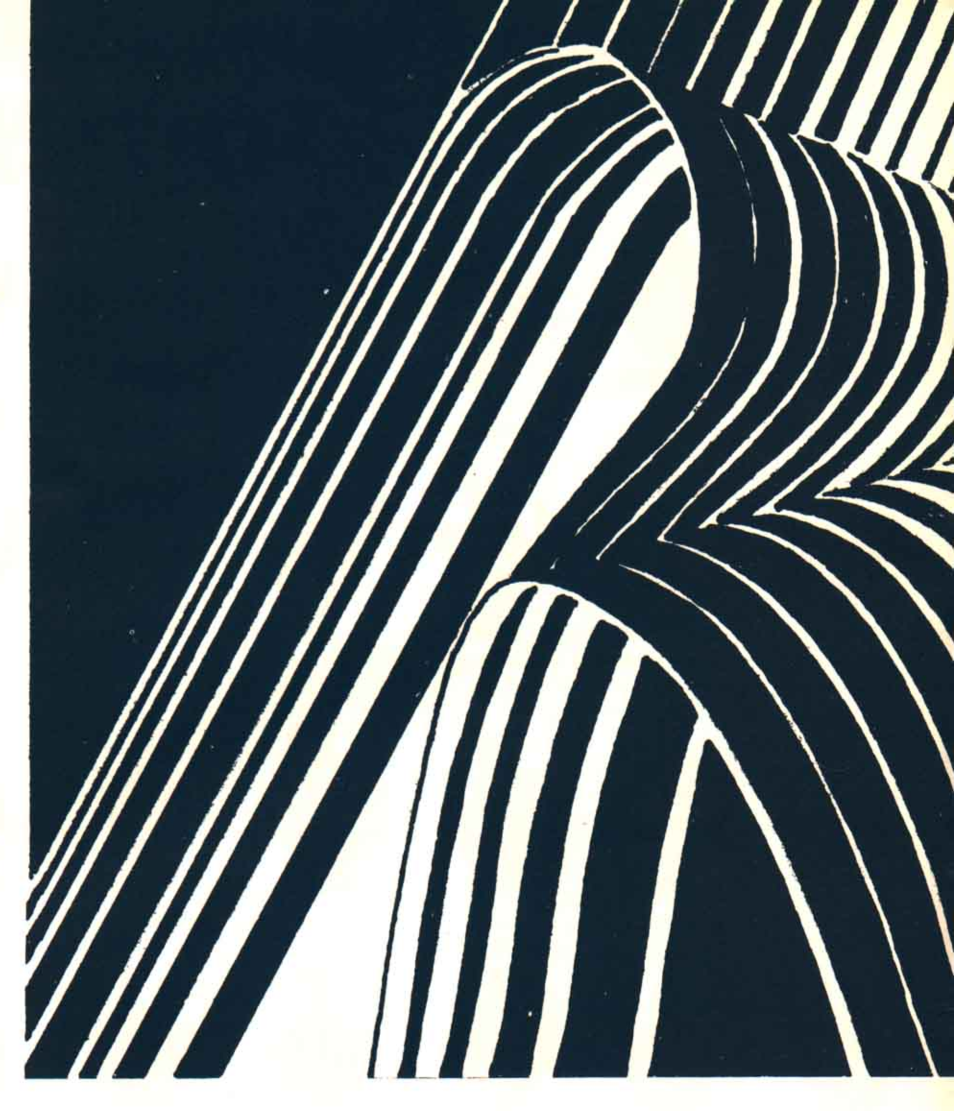


Ordenador Doméstico

HIT BIT

SONY

Para lo que guste ordenar. MSX



El código máquina: ese desconocido

(cap.2)

HEmos visto que el microprocesador manejaba un "libro" que denominamos memoria. Esta es un tanto especial, ya que en cada "hoja" solo cabe lo que se denomina un byte, por lo que para escribir más de uno deben irse poniendo en memorias páginas) sucesivas.

El sistema que utiliza el "robot", llamado Z-80, para acceder a una memoria se basa en que todas están numeradas secuencialmente a partir de cero y no hay dos hoja "bytes de memoria" con la misma numeración. De modo que para acceder a una, da un número de página y automáticamente ésta le aparece a la vista. En el Z-80 la mayor página que puede manejar es la 65535 (FFFFh) lo que hace un total de 65536 hojas (bytes) manejables (recordemos que se empieza a numerar a partir de cero) divididas entre ROM y RAM. Aunque algunos modelos de ordenadores no tengan tanta instalada y no sea manejable, ya que aunque sepa pronunciar el número de una hoja, no puede ver su contenido si el libro se acaba antes.

Otro detalle a tener en cuenta es que en una memoria, debido a la construcción física del ordenador, siempre hay algún número, aunque sea un cero. De modo que siempre que miremos si en una memoria hay un número veremos si lo hay. Otra cosa es que éste sea significativo o *"garbage"*, que es un término inglés que se utiliza para referirse a la basura y que en este caso significa cualquier dato que no sea de ninguna utilidad. En realidad, al encender el ordenador cada memoria contiene un número cuyo valor es indeterminado y que es usado como si fuese *"garbage"*.

La señal del RESET

Recordarán ustedes que en el capítulo anterior comparábamos al Z-80 con un robot que entendía una serie de instrucciones, en base a las cuales hacia todo lo que le mandábamos. Pero aquellos que hayan pensado en éste tema habrán descubierto un 'error' en la explicación, ya que no se sabía que instrucción



iba a ser la primera que ejecutase el robot, cuando lo pusiésemos en marcha. ¿Quizás la primera que haya en el libro (memoria)?, ¿o empezaría por el final?. ¿Tal vez empieza en cualquier lugar de la memoria según le da al encenderse?. Este último caso, que puede ser peliagudo si se quiere hacer un programa no es, por suerte, el real. En realidad lo que hace al encenderlo es ejecutar a partir del primer 'byte'. Pero los ordenadores MSX tienen en esta memoria una instrucción "DI" que veremos más adelante y que anula las interrupciones y luego un "JUMP" o salto a otra dirección de memoria, según se indica en el número que viene a continuación de éste "JUMP". De modo que si éste es un 1800, empezará a ejecutar en la 1800. Este modo de direccionamiento es bastante potente y nos permite colocar el programa en cualquier parte de la memoria, con la única condición de que al inicio de ésta se encuentre almacenado el número de inicio del programa.

Los ordenadores MSX hacen esto automáticamente cuando los encendemos y empiezan la ejecución del BASIC por lo que este proceso es invisible para el usuario. Si lo hemos explicado aquí, ha sido para comprender que el microprocesador siempre tiene una ejecución defintiva y controlable y que, por tanto, podremos dirigirle a nuestras propias rutinas.

De bytes a palabras y de palabras a bytes

Recordarán que dijimos que nuestro ordenador tenía la memoria agrupada en bytes. Así, en la primera de las 'páginas' de nuestro libro ficticio (que es la memoria), cabía un byte (8 bits), en la segunda otro, etc. Para poder almacenar una "palabra" o número de 16 bits hay que dividirla en dos y almacenar cada parte en un byte. Este proceso de división resulta muy sencillo en binario o hexadecimal, ya que basta partir el número por la mitad según se le mira. Por ejemplo '1011001110011101b' que es una palabra, se partirá por la mitad dándonos como valor superior '10110011b' y como valor inferior '10011101b' (recuerde que la "b" que va detrás del número indica que es binario). En hexadecimal el proceso es similar, para dividir la palabra 'B39Dh' en dos basta coger los dos digitos de la izquierda, 'B3h' para tener la parte superior y los dos de la derecha, '9Dh', para la parte inferior.

En decimal el proceso es algo más complicado y hay que dividir 45961 entre 256. El resto, 179, es la parte inferior y el resultado (la parte entera) es la parte superior que, en este caso, es 157. La forma en que se almacenan dentro del ordenador es la contraria de la que se puede pensar en un principio, ya que la parte inferior se almacena primero y lo superior va a continuación. Esto hace que si nos imaginamos la memoria, el número se vea al revés de lo que esperámos. En la figura 1 se muestra la palabra de 16 bits en hexadecimal y su almacenamiento en dos memoria que hemos denominado genéricamente 'nnnn+1' que se suponen consecutivas.

Nuestra primera instrucción

Después de tanto ajetreo, vamos, por fin, a aprender la primera

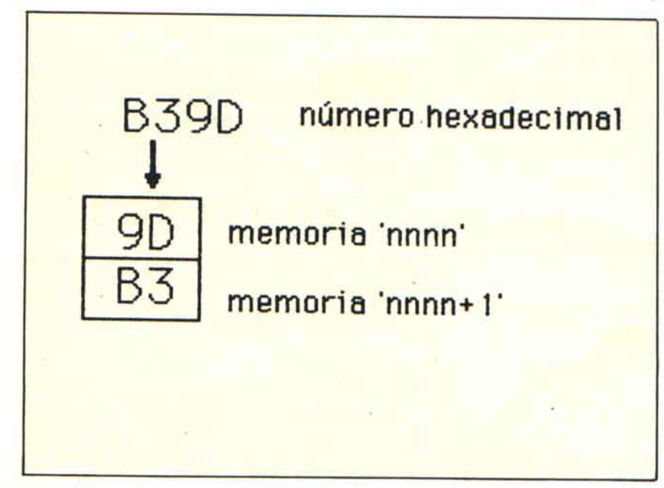


FIGURA 1

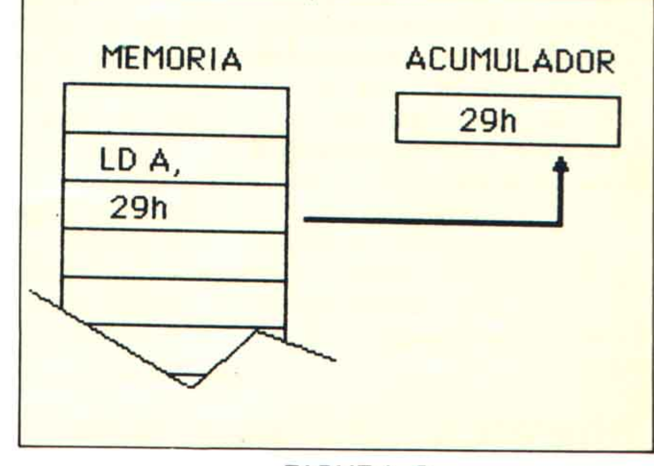


FIGURA 2

PHILIPS MSX



El sistema más sabio

PHILIPS introduce en España el HOMECOMPUTER más sabio, el sistema MSX, nuevo estandard mundial.

¡Con cuanta sabiduría se ha pensado en cada una de sus características!

Con el PHILIPS MSX puede realizar mil combinaciones de elementos: monitores, impresoras, floppys, programas educativos, de juegos y aplicaciones profesionales, gracias a su compatibilidad total tanto en hardware como en software.

El PHILIPS MSX está tan sabiamente diseñado que Vd. puede elegir entre conectarlo al televisor de su casa, o a un monitor monocromo o de color.

De igual modo puede utilizar como unidad de almacenamiento de memoria un cassette normal o un Floppy Disc del sistema MSX.

¡Y qué potencia tiene el PHILIPS MSX!

Es tanta, que si lo utilizamos con un Floppy Disc y junto a MSX-DOS, es compatible con sistemas de tipo profesional y de precio mucho más elevado.

Y aquí no acaba la sabiduría con que ha sido creado el PHILIPS MSX.

Puede hacerlo crecer según sus necesidades, desde un sencillo ordenador doméstico, con el lenguaje Basic más potente del mercado, hasta un sistema de tipo profesional que puede llegar a una capacidad máxima de 1.024 K bytes.

PHILIPS MSX. Nunca se le quedará pequeño, nunca se le quedará anticuado.

PHILIPS MSX, creado como un equipo atractivo, fácil de usar y muy asequible de comprar.

¡PHILIPS MSX, sin duda, el sistema más sabio!

MSX-DOS es compatible con CP/M™ y posee la misma estructura de ficheros que MS-DOS™.

Todos los sistemas MSX son compatibles entre sí.

MSX, MSX-DOS™ y MS-DOS™ son marcas registradas de Microsof Corp.
CP/M™ es una marca registrada de Digital Research.

Si desea algún tipo de información relacionada con el campo del HOMECOMPUTER, estamos a su disposición en el teléfono

(91) 413 22 46

Desearía recibir más información sobre el PHILIPS MSX.

Nombre ....

Apellidos ....

Domicilio ....

....

PHILIPS IBERICA S.A.E.
Apartado de Correos 50.800
28080 MADRID

PHILIPS MSX HOMECOMPUTER SYSTEM

El amigo sabio de la familia.

instrucción del Z-80. Esta se denomina "LD A", y sirve para meter un número dentro del acumulador. Este, como vimos en el artículo anterior, servía para hacer operaciones con los números,pero antes de operar es necesario, evidentemente, tener el dato en cuestión, del mismo modo que antes de hacer una operación en una calculadora debemos pulsar el número con el que queremos operar (y si son dos debemos teclar primero antes de dar a la operación y al segundo). El objetivo de esta instrucción es, por tanto, meter en el acu mulador el número con el que vamos a operar.

La segunda es de dónde sacamos el número. Evidentemente no puede ser el acumuladdor, ya que sería una perogrullada sacar un número de él para volverlo a meter. Podría ser de cualquiera de otros registros o, también, de la memoria. En realidad puede ser de cualquiera de estos dos últimos pero, además, cuando cargamos de la memoria podemos hacerlo de muy diversos modos que se denomiará 'modos de direccionamiento'. El primero que vamos a ver es el direcionamiento inmediato.

Este tipo de direccionamiento el número que viene después de la instrucción es el que se carga en el acumulador. En la figura 2 se ve un ejemplo de esto en el que se hace "LD A, 29h", con los que se obliga al

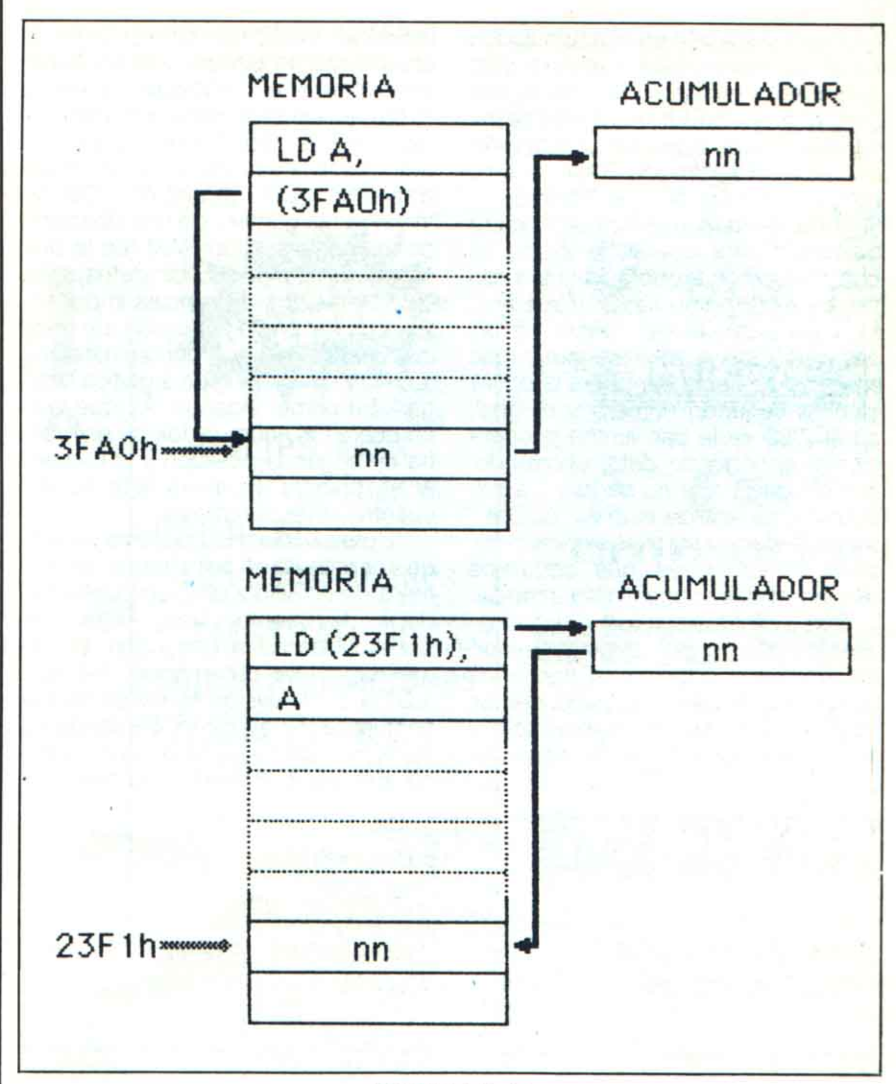


FIGURA 3

DIRECCIONAMIENTO	INMEDIATO	DIRECTO
LD A,	62	58
LD....,A	no hay	50
ADD A,...	198	no hay

FIGURA 4

microprocesador a guardar el 29h en el acumulador. Así mismo, "LD A, 54h" guardaría 54h en el acumulador. Evidentemente, éste número sólo puede estar comprendido en el rango 0-255 (o Oh-FFh que es lo mismo en hexadecimal) ya que el acumulador es un registro de 8 bits y este rango es el máximo que admite)

Una vez que tenemos un número dispuesto para operar, lo lógico es que realizemos la operación. Para ello vamos a ver como se hace la suma. En este caso, el uso difiere de las calculadoras, ya que mientras que en estas se debía dar ahora la operación, el segundo número y el igual, en el Z-80 se le dan juntos la operación y el segundo dato, ahorrándonos el 'igual', que no se usa. La instrucción que realiza esto es "ADD A," a la que debe seguir el segundo número a sumar, ya que seguimos usando el direccionamiento inmediato (los demás tipos de direccionamiento los iremos viendo posteriormente). De modo que si habíamos puesto "LD A, 29h" y ahora ponemos "ADD A, 47h", se nos sumará 29h y 47h, quedando 71h en el acumulador.

Pero cuando tenemos el resultado es necesario guardarlo, ya que si hacemos cualquier operación en el acumulador lo borrará y si no lo hemos metido en memoria no habrá forma de volver a usarlo. Para ello vamos usar la misma instrucción "LD" que vimos antes pero de un modo distintos: "LD (nnnn),A". Donde 'nnnn' es el número de una dirección de memoria del tipo RAM (en la que se puede leer y escribir datos). Vemos ahora dos diferencias importantes con la primera versión de esta instrucción (que denominaremos 'LOAD', que en inglés significa 'cargar'. En primer lugar, la 'A', que indica que es el acumulador, se encuentra en segunda posición y el número en la primera. Además éste se encuentra entre paréntesis.

Como sabemos que este segundo formato sirve para pasar un número de un sitio a otro, se puede deducir facilmente una regla de construcción de la instrucción. En Primer lugar, se debe poner siempre "LD", a continuación se indica donde se guarda y, por último, de donde se saca. Por tanto, "LD xxxx, yyyy" ordena que hay que pasar el número que se señala en 'yyyy' adonde indica 'xxxx'. Donde 'yyyy' y 'xxxx' pueden ser registros del microprocesador, posiciones de memoria y otras combinaciones que veremos luego.

También vimos que en la instrucción "LD (nnnn),A, el número iba entre paréntesis cosa que no sucedía al principio. Esto se usa para indicar otro tipo de direccionamiento, ya que en este caso no es válido el direccionamiento inmediato porque el primer número (xxxx) debe indicar la dirección donde se almacena y en el direccionamiento inmediato no se indica ninguna dirección sino el número en si. En este caso, estamos usando lo que se denomina direccionamiento directo. En este tipo de direccionamiento el número se pone entre paréntesis e indica, no ya el dato como en el caso anterior, sino la dirección donde se almacena el dato. Es decir, la instrucción "LD" (23F1h),A, almacenará el contenido del acumulador en la dirección 23F1h, del mismo modo "LD A, (3FA0h)" no meterá 3FA0h en el acumulador, sino el contenido de la memoria EFA0h. Estos dos tipos de direccionamiento han sido representados en la figura 3 y se puede ver la diferencia entre ambos.

Con estas instrucciones que hemos explicado ya tenemos la base para construir un sencillo programa que sume dos números y almacene su resultado en un memoria. Si queremos sumar 25h y A7h almacenando el resultado en la memoria 40500, habría que poner:

```
LD A, 25h
ADD A,A7h
LD (40500),A
```

Pero aún nos faltan por fijar un par de cosas más. Hay que tener en cuenta que en la memoria sólo se pueden almacenar números, pero lo que hemos estado poniendo hasta ahora son palabras. Evidentemente, estas no las podemos meter directamente a la memoria, sino que hay que poner números que equivalen a lo mismo. La razón de que hayamos estado tratando con palabras (denominadas mnemotécnicos) se debe a que es mucho más comprensible decir "LD A," o "ADD A," que 206. Estos últimos números son los

MNEMOTECNICO		CODIGO	COMENTARIO
LD	A,25h	62 37	(62 LD A, 37 es 25h)
ADD	A,A7h	198 167	(192 es ADD A, 197 es A7h)
LD	(26000),A	50 52 158	(50 es LD, 144+101*255 es 40500)
RET		201	(201 es RET)

FIGURA 5



que tenemos que meter y se denominan códigos de operación.

Existen unos programas especiales denominados ensabladores que permiten que se tecleen las instrucciones con mnemotécnicos y ellos las transforman automáticamente a los códigos correspondientes, pero al no estar demasiado difundidos los ensambladores para los MSX, a partir de ahora explicaremos los mnemotécnicos y los códigos correspondientes.

El número correspondiente a un código de operación no solo varía con ésta (el de "LD A," es distinto al "ADD A"), sino también con el modo de direccionamiento o los registros que estemos usando. De modo que "LD A," tiene uno distinto a "LD (xxxx), A. En la tabla de la figura 4 se dan los códigos correspondientes a las instrucciones que hemos visto, según se utilice un direccionamiento u otro.

Por otra parte, hay que considerar también que si dejamos el programa tal como está, después de ejecutar la última instrucción seguirá ejecutando lo que encuentre en las siguientes posiciones de memoria que, aunque nosotros no hayamos metido nada, tendrá almacenando un valor aleatorio.

Para comprender como pasa esto, pensemos que al encender nuestro ordenador, se creará por si solo, un programa BASIC que ocupase toda la memoria. Si nosotros metiesemos uno nuestro al princípio, cuando le mandásemos ejecutar, ejecutaría el nuestro y luego seguiría con lo que tuviese en memoria a menos que lo pusiésemos END o un STOP al final de nuestro programa. Aunque esto no sucede en BASIC, si que pasa en lenguaje máquina y hay



que evitar que suceda, ya que no sabemos como entenderá el Z-80 estos números y, muy posiblemente, se "colgará" quedandose bloqueado y sin permitirnos hacer nada.

Teniendo en cuenta que nosotros lo vamos a llamar desde el BASIC con un USR (si no conoce el significado exacto de esta instrucción BASIC, le diremos que lo que hace es llamar a una subrutina en lenguaje máquina en la posición que hayamos definido previamente con un DEF USR) que tiene un funcionamiento parecido al GOSUB del BASIC pero con la subrutina en lenguaje máquina, habrá que poner al final el equivalente del RETURN pero, evidentemente, en lenguaje máquina. Esta instrucción es la "RET" que hace que se devuelva el control al sitio desde donde se ha llamado a la rutina. Esta operación no tiene ningún tipo de direccionamiento, ya que este sólo se usa con los datos extras necesarios, pero en este caso no hay ninguno y el código de operación es siempre C9h (201). Nuestro programa queda como se indica en la figura 5, en la que se ha metido el listado con mnemotécnicos y al lado los correspodientes números (en decimal).

De este listado podemos ver que toda la instrucción que lleve datos anexos, como son LD y ADD, se escribe poniendo primero el código de la operación y luego el dato. Si éste es de 16 bits (como la instrucción LD (xxxx),A) se escribe la parte inferior primero, como se explicó anteriormente.

Una vez que tenemos los códigos numéricos correspondientes, hay que meterlos en memoria. Aquellos que dispongan de un ensamblador no tendrán problemas, ya que éste realiza la operación automáticamente, pero si no, se debe recurrir a un cargador hexadecimal. Este programa, cuyo listado se dá en la figura 6, sirve para introducir rápidamente y con el menos número de fallos posible los datos en memoria. Se ve en su listado que en la línea 1000, existe una instrucción DATA. A continuación de ella (y en líneas posteriores si hiciese falta) se deben meter los códigos poniendo al terminar un 256 como último dato. Este número es ilegal como código de instrucción (el máximo admitido es el que cabe en un byte, es decir, 255) y por esta misma razón sirve para indicar el final de los datos al programa. Una vez hecho esto ponemos en marcha el programa tecleando RUN y cuando nos pregunte "DIRECCION INICIAL": se le contesta el número de memoria a partir de la cual queremos almacenar el programa. En nuestro caso esta es la 40000, por lo que deberemos teclearlo así. Esta dirección sólo puede contener "garbage" (al igual que la 40500 y posteriores) por lo que se puede usar sin miedo a destruir ningún dato importante.

Si usted prefiere puede ponerlo en otra dirección distinta, pero hay que tener en cuenta que esta debe ser una direción existente en RAM, para que podamos meter el programa (si es ROM podemos leer pero no escribir y sólo sirve para almacenar los programas que el ordenador trae de fábrica, no los nuestros) y que no sea usada por el sistema para almacenar sus cosas (los programas BASIC y otros datos internos del ordenador).

Una vez introducido el programa hay que ejecutar dos instucciones en BASIC. La primera es:

```
DEFUSRO=40000
```

Que asigna al 'vector' de llamada 0, el número 40000. La segunda, que es la que propiamente realiza la llamada, se escribe:

```
PRINT USRO (1)
```

Donde el cero hace referencia al vector de llamada, haciendo que se salte a la dirección de llamada que asignamos a este (40000). Si hubiésemos puesto DEFUSRO=1000, el salto sería a la mil. El uno de esta última instrucción es un parámetro que se puede pasar a nuestra subrutina, aunque ahora no lo usamos y entre los paréntesis se puede poner cualquier número que se quiera. Cuando hayamos ejecutado estas dos instrucciones veremos que en pantalla nos pone "OK". Para averiguar si ha sumado bién hay que sacar el contenido de la memoria 40500, que es donde se almacenó el resultado. Para que nos lo muestre

```
10 INPUT "DIRECCION INICIAL:";D
20 READ A
30 IF A=256 THEN PRINT "TERMINE":END
40 POKE D,A
50 D=D+1
60 GOTO 20
1000 DATA 62,37,198,167,50,52,158,201,256
```

FIGURA 6

por pantalla recurriremos a la función PEEK, que devuelve el contenido de la memoria que se indique. De este modo PRINT PEEK (40500) nos mostrará el contenido de la memoria 40500.

El resultado es, maravilla, 204 (CCh) como era de esperar.

Modificando nuestro programa.

El programa que hemos descrito previamente tiene algunos defectos, imaginaremos que ahora queremos sumar otros dos números distintos. Esto nos obligaría a volver a escribir el programa de nuevo cambiando los "LD" y "ADD" para meter los nuevos números. Esto se debía a que utilizábamos el direccionamiento inmediato que nos obliga a este procedimiento.

LD (40500),A

40500	resultado
40501	segundo núm.
40502	...
	...

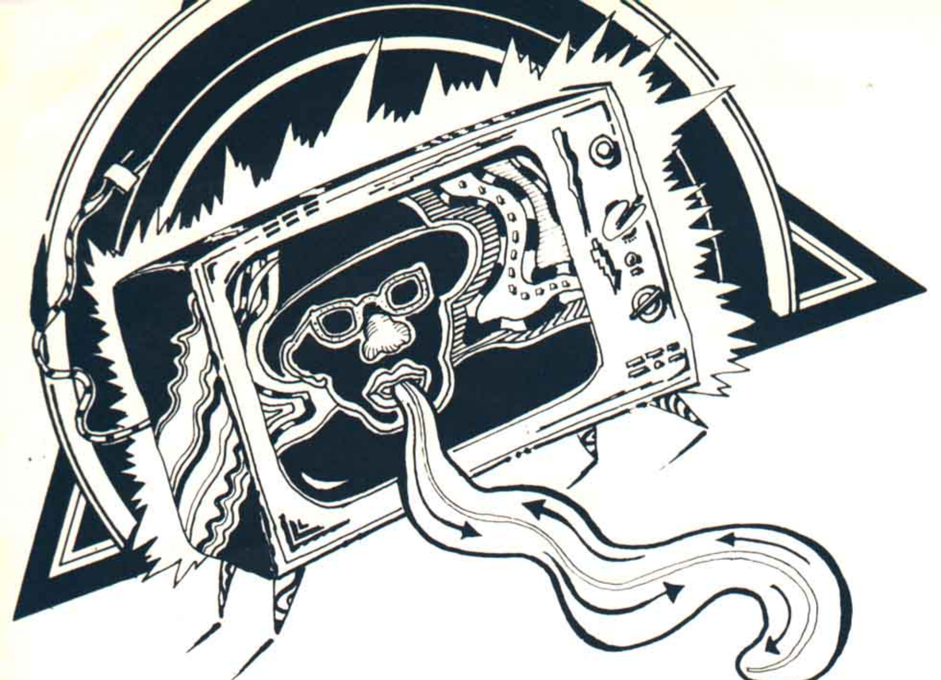
LD (40502),A

40500	primer núm.
40501	segundo núm.
40502	resultado
	...

FIGURA 7

Pero hemos aprendido a usar otro tipo de direccionamiento que es el directo, vamos a hacer que nuestro programa sea más versatil haciendo uso de este direccionamiento.

En primer lugar ponemos "LD A, (40500)" que, como vimos al hablar de este tipo de direccionamiento,



nos mete en el acumulador, no un 40500 (que además no cabría) sino el que está almacenado en la dirección 40500. Un proceso similar se debería usar con el ADD, poniendo "ADD A, (40501)" para que nos sumara al acumulador el contenido de la memoria 40501. Pero nos encontra mos con el problema de que no existe este tipo de direccionamiento con la instrucción ADD. Pero en cambio existe otro, que se denomina indirecto, en el que la dirección de donde queremos sacar el número no se pone a continuación sino que se toma del par de registros HL, cosiderándolos como uno de 16 bits, el formato es "LD A, (HL)" donde los paréntesis indican que en HL está guardada la dirección y no el número. Evidentemente esto implica que se debe cargar previamente esta dirección en este par de registros, lo que hace con la instrucción "LD HL, 40001", que obliga al microprocesador a meter el número 40001 en los registros HL como si fuese uno solo de 16 bits.

El código de operación de "LD HL, nnnn" es 33 y debe ir seguido por dos bytes que indiquen el número a cargar en el formato de la parte inferior primero. Por otra parte, el código de la instrucción "LD A, (HL)" es 134 y no lleva datos ya que todos los necesarios se indican con la misma instrucción.

Con esto conseguimos que el programa sea siempre el mismo y variamos solamente el contenido de las memorias 40500 y 40501 y no el programa que se encuentra alamacenado a partir de la 40000.

Por último, la instrucción que guarda el resultado de la suma, que era "LD (40500),A", la vamos a cambiar por "LD (40502),A" de modo que se almacene a continuación de los dos números que introdujimos. En realidad podiamos haberla dejado como estaba, pero hubiera borrado el primer número que introdujimos, de este modo, en cambio, al final tenemos los tres. Por último, como la vez anterior, hemos de poner "RET". La diferencia entre un caso y otro se puede ver en la figura 7 donde se ve como quedarían estas memorias si dejásemos 40500 y como quedan al poner 40502. El listado de esta versión, junto con sus equivalentes decimales, se ve en la figura 8, siendo estos números los que se deben teclear ahora en las sentencias DATA del programa cargador hexadecimal.

Para usar esta versión del programa, antes de llamar a nuestra rutina por medio del USR, hemos de introducir los datos en las memorias correspondientes por medio de una instrucción POKE del BASIC, que esta es la única instrucción que tenemos para acceder directamente a las posiciones de memoria desde el BASIC, al no permitirnos el ordenador trabajar directamente en lenguaje máquina. El formato de esta instrucción es:

```
POKE dddd,nn
```

Donde 'dddd' es la dirección (en decimal) en la que queremos almacenar el número y 'nn' es el número que queremos almacenar.

Por ejemplo, para numerar 25h y A7h, debemos coger el valor decimal de ambos números (37 y 167 respectivamente) y hacer los siguientes pokes:

```
POKE 40500,37
POKE 40501,167
```

Y después procederemos a llamar a nuestra rutina de suma por medio de USR como vimo anteriormente. Del mismo modo, al acabar nos devuelve el mensaje "OK" y para ver el resultado de nuestro programa haremos.

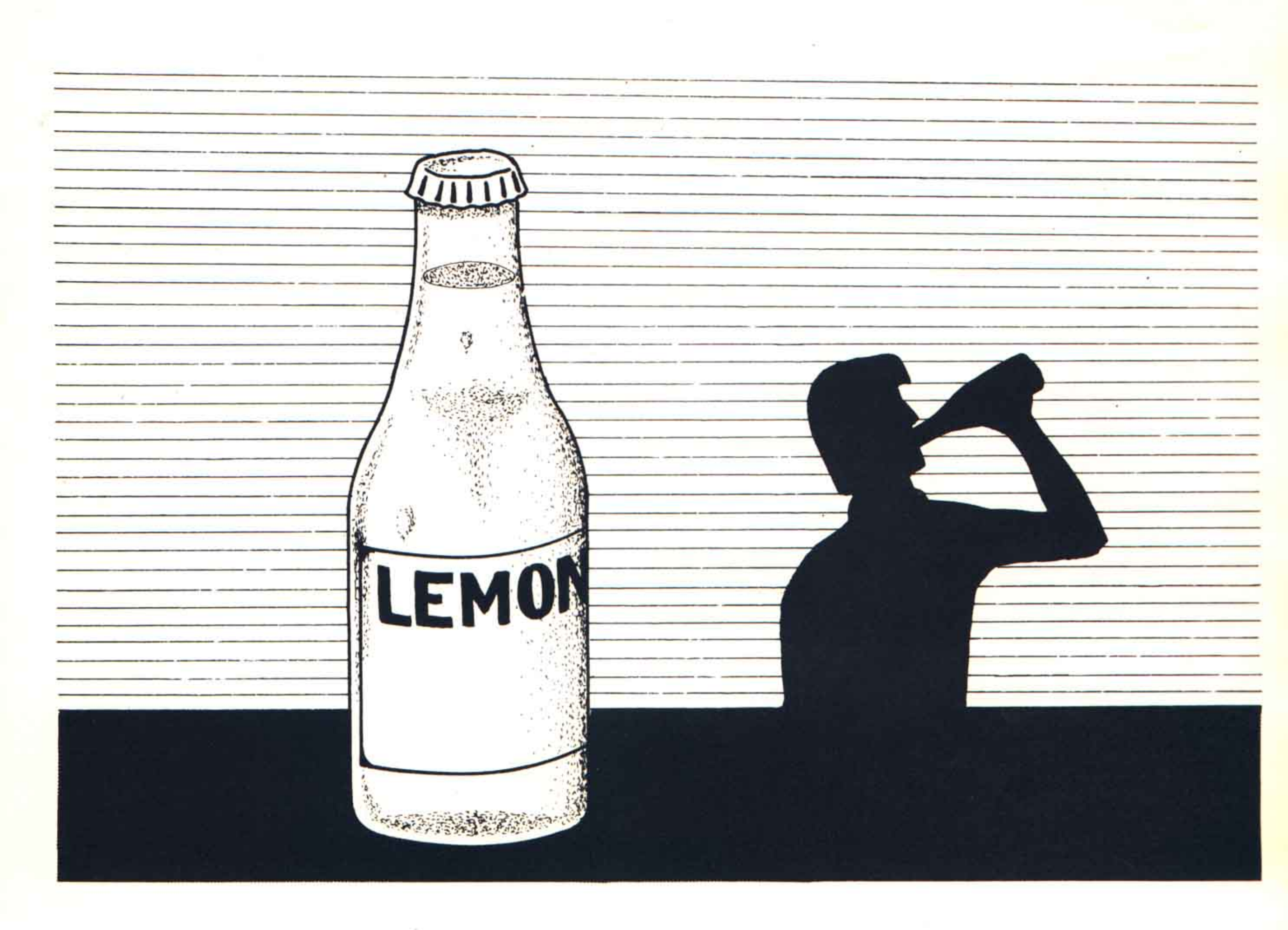
```
PRINT PEEK (40502)
```

Obtenido el resultado de la suma.

Hay que tener en cuenta otro detalle importante a la hora de suma. Dijimos anteriormente que el número mayor que cabe en el acumulador es el 255, por lo que si el resultado de la suma lo supera, se producirá lo que se denomina 'acarreo', quedando en este registro el resto a dividir entre 255. Tomemos, por ejemplo la suma entre 184 y 95 (ambos en decimal), su suma es 279, y el resto al dividir entre 255 y 24, que es lo que se almacenará en el acumulador y el resultado que nosotros veremos, que será falso. Pero no se preocupe, existen métodos para hacer que se sumen números más largos, coo veremos en el prósimo capítulo.

MNEMOTECNICO		CODIGO			COMENTARIO
LD	A,(40000)	58	52	158	(58 es LD A, 64+156*256 es 40000)
LD	HL,40001	33	53	158	(33 es LD HL, 65+156*256 es 40001)
ADD	A,(HL)	134			(134 es ADD A,(HL))
LD	(40002),A	50	54	158	(50 es LD, 144+101*255 es 40502)
RET		201			(201 es RET)

FIGURA 8



Limonada

Todos en algún momento nos hemos sentido con ganas de ejercer de ejecutivo agresivo. De una forma u otra, existen juegos en el mercado que te permiten desarrollar esa capacidad.

Este programa es una versión más tractiva, ya que la actividad se desarrolla en la playa de Villalimones, donde eres dueño de un kiosko de limonadas. Como todo buen comerciante, deberás saber cuando tendrás que intervenir en Publicidad número de botellas a comprar y precio de mercancia, que en este caso es el vaso de limonda, para obtener el máximo beneficio posible.

Claro que todo no depende del número de carteles expuestos, ni del precio del vaso. Sabemos que un factor importante a la hora de poner cualquier chiringuito en la playa es el tiempo, el factor ambiental. Es decir, que si hace frio, podremos perder mucho dinero. Además como todo juego, el factor ambiental, que en éste caso puede ser, tiempo soleado, lluvioso o incluso que haya nieve, se nos notificará al terminar de introducir todos los valores indicados anteriormente. Por lo que si somos demasiados optimistas y nos encontramos con tiempo lluvioso podremos perder dinero. Si tenemos suerte y en lugar de llover, hace sol, perderemos bastante dinero... o no.

En este juego, podrán participar hasta 4 jugadores que se enfrentarán hasta un total de 14 días. Estas dos opciones son las primeras en aparecer cuando ejecutamos el programa. A continuación se mostrarán una serie de mensajes a seguir y a contestar, de manera que al final del juego y en base a las intervenciones realizadas, podremos saber el beneficio o pérdida que hemos tenido. Es fácil alterar el programa para que de alguna forma o otra puedan aparecer gráficos o alguna musiquilla, esto lo dejamos en manos de los lectores para que vayan practicando.

```
10 KEYOFF:CLS
20 LOCATE 8,5: PRINT"L I M O N A D A"
30 LOCATE 8,6: PRINT"==============="
40 LOCATE 0,10: PRINT"Necesitas instrucciones (s/n)"
50 LET A$=INKEY$
60 IF A$="S" OR A$="s" THEN GOTO 1180
70 IF A$="N" OR A$="n" THEN GOTO 90
80 GOTO 50
90 CLS: LOCATE 0,3: PRINT "Cuantos jugadores (2-4)";:INPUT P
100 IF P<2 OR P>4 THEN GOTO 90
110 DIM M(P),A(P),L(P),B(P),Q(P),I(P),S(P)
120 CLS: LOCATE 0,3: PRINT "Cuantos días jugaremos (max 14)";:INPUT AD
130 IF AD<1 OR AD>14 THEN GOTO 120
140 FOR F=1 TO P
150 M(F)=25
160 A(F)=0
170 R(F)=0
180 B(F)=0
190 NEXT F
200 CLS
210 FOR E=1 TO AD
220 PRINT "L I M O N A D A         DIA";E:PRINT "---------------"
230 W=1+INT(3*RND(-TIME))
240 IF W=0 OR W=3 THEN W$="SOL"
250 IF W=1 THEN W$="LLUVIA"
260 IF W=2 THEN W$="NIEVE"
270 LOCATE 0,4: PRINT "Informe metereológico de Villalimones"
280 LOCATE 0,6: PRINT W$
290 LOCATE 0,20:PRINT"Pulsa una tecla para continuar"
300 IF INKEY$ ="" THEN 300
310 AP=1+INT(3*RND(-TIME))
320 BP=INT(1+INT(4*RND(-TIME))*.5)
330 FOR F=1 TO P
340 CLS
350 PRINT"L I M O N A D A"
360 PRINT"---------------":PRINT
370 PRINT"Jugador ";F
380 PRINT"Saldo $";M(F)
390 PRINT"Anuncios ?, Cada uno cuesta $";AP: INPUT A(F)
400 PRINT A(F)
410 PRINT: PRINT"Cuantas botellas quieres?
420 PRINT"Cada una cuesta $";BP:PRINT"y tienes para 10 vasos";:INPUT B(F)
430 PRINT B(F)
440 M(F)=M(F)-(B(F)*BP)-(A(F)*AP)
450 PRINT:PRINT"Tu saldo es de $";M(F)
460 PRINT"Precio de venta por vaso";:INPUT L(F)
470 L(F)= INT(ABS(L(F)))
480 PRINT L(F)
490 LOCATE 0,20:PRINT"Pulsa una tecla para continuar"
500 IF INKEY$="" THEN 500
510 NEXT F
520 IF W$="SOL" THEN G=175
530 IF W$="LLUVIA" THEN G=75
540 IF W$="NIEVE" THEN G=10
550 CLS:PRINT"L I M O N A D A    DIA  ";E:PRINT"---------------"
560 PRINT:PRINT W$
570 V=INT(RND(-TIME)*50)+G
580 PRINT:PRINT "Hoy has tenido ";V;" clientes"
590 LOCATE 0,20:PRINT"Pulsa una tecla para continuar"
600 IF INKEY$="" THEN 600
610 CLS
620 FOR C=1 TO P
630 Q(C)=(B(C)*10)
640 I=V*10
```

```
650 S(C)=INT(S(C)*.75)
660 S(C)=I/L(C)
670 IF L(C)>30 AND L(C)<=50 THEN S(C)= INT(RND(-TIME)*(V/(V-2)))
680 IF L(C)>50 THEN S(C)=0
690 Z=(RND(-TIME)*(A(C)*5))
700 S(C)=S(C)+(Z*2)
710 NEXT C
720 N=0
730 FOR O=1 TO P
740 N=N+S(O)
750 NEXT O
760 IF N>V THEN M=N-V
770 IF N>V THEN M=INT(M/P)
780 IF N>V THEN FOR Y=1 TO P
790 IF N>V THEN S(Y)=S(Y)-M
800 IF N>V THEN NEXT Y
810 FOR H=1 TO P
820 IF S(H)>Q(H) THEN S(H)=Q(H)
830 IF S(H)<0 THEN S(H)=0
840 IF S(H)>V THEN S(H)=V
850 NEXT H
860 CLS
870 FOR J=1 TO P
880 PRINT "L I M O N A D A":PRINT "---------------"
890 MY=INT(S(J)*L(J))/100
900 M(J)=M(J)+MY
910 PRINT "Jugador ";J;" Tu saldo es $";M(J)
920 PRINT "Has vendido ";INT(S(J));" vasos."
930 PRINT "Tenías limonada para ";Q(J);" vasos"
940 PRINT "Precio de cada vaso ";L(J)
950 PRINT "Anuncio(-s) ";A(J)
960 PRINT "Beneficios $";MY-BP*B(J)-AP*A(J)
970 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
980 IF INKEY$="" THEN 980
990 CLS
1000 NEXT J
1010 NEXT E
1020 CLS
1030 PRINT"L I M O N A D A"
1040 PRINT"---------------
1050 PRINT:PRINT "FIN DE LA TEMPORADA":PRINT
1060 FOR U=1 TO P
1070 PRINT "Jugador ";U
1080 PRINT "Tu saldo ha sido de $";M(U)
1090 PRINT
1100 NEXT U
1110 LOCATE 0,20: PRINT "Pulsa una tecla para continuar"
1120 IF INKEY$="" THEN 1120
1130 CLS
1140 PRINT "Otra partida (s/n)";:INPUT X$
1150 IF X$="s" OR X$="S" THEN 120
1160 IF X$="n" OR X$="N" THEN END
1170 GOTO1170
1180 CLS:LOCATE 8,0:PRINT "L I M O N A D A":LOCATE 8,1:PRINT "---------------"
1190 PRINT:PRINT "El objetivo del juego es ganar el    máximo dinero posible."
1200 PRINT "Para ello dispones de un kiosko de   limonada en la playa de Villali
mones."
1210 PRINT "Cuentas con $25, que tendrás que in- vertir en botellas y publicidad
."
1220 PRINT "En base a lo que inviertas ganarás   dinero o no, eso depende del ti
empo  y de la suerte . . ."
1230 LOCATE 0,20: PRINT "Pulsa una tecla para continuar"
1240 IF INKEY$="" THEN 1240
1250 GOTO 90
```

Simulador de vuelo



Si te gustan las emociones fuertes y tienes el pulso estable, prueba a ejecutar este programa, que por cierto es muy apto para cardíacos, ya que se necesita una buena dosis de paciencia para hacer volar el aparato.. Se trata de un simple simulador de vuelo, pero con todos los alicientes de los simuladores más serios. Podríamos decir que se trata de una versión para "andar por casa".

Tendrás que elegir entre dos aeropuertos para aterrizar o despegar (si escoges la segunda opción de las tres que aparece al ejecutar el programa), con la representación del terreno por separado y un mapa a pequeña escala de la zona. Iniciada la maniobra de aproximación, aparecerá ante nosotros la pista de aterrizaje, siempre y cuando estemos por debajo de los 2.000 metros. Asimismo, el mapa del terreno se visualizará cuando despeguemos. Tu posición se mostrará mediante un pequeño avión destellante, pudiendo controlar la dirección del vuelo a través de los relojes dedicados especialmente para tal fin, que es el **HDG.** En la parte inferior de la pantalla, podrás ver el panel de instrumentos, que serán la principal ayuda en el vuelo.

Estos son muy útiles a la hora de conocer el estudio del avión, la ruta a seguir, la potencia de los motores, etcétera. Estos instrumentos son los siguientes:

HDG.— Indica la dirección de nuestro avión. El Norte está a 0 grados, el Este a 90 grados, el Sur a 180 y el Oeste a 270 grados.
RDR.— Es la posición del timón de cola, si es negativo indicará giro a la izquierda, y si es positivo hacia la derecha.
GAS.— La cantidad de fuel que nos queda.
VEL.— Indica la velocidad del avión.
POT.— La potencia en cada momento.
ROC.— Velocidad de ascenso o descenso de nuestro aparato. La luz roja indica que estamos descendiendo a la velocidad que indique este reloj.
FLP.— Con este reloj, comprobaremos la posición de los falps, o cuando está totalmente cerrado y 30 cuando está totalmente abierto.
TREN ARRIba o ABAJo.— Indica la posición del tren de aterrizaje. El cuadro que aparece a la izquierda del panel de instrumentos es un horizonte artificial, que nos ayudará a comprobar la horizontalidad de nuestro avión. Cuando la línea roja esté por debajo de la mitad del cuadro, es que estamos subiendo, mientras que si está por debajo de la mitad, indicará que estamos bajando. De la misma forma podremos comprobar el ángulo de giro, ya que cuando la línea roja vaya desde la esquina inferior izquierda hasta la esquina superior derecha, es que el avión se está inclinando hacia la izquierda.

El avión se controla con las teclas E, S, D y X, donde cada tecla tiene una función que es la siguiente: la X para subir, la E para bajar, la D para girar hacia la izquierda y la S para girar hacia la derecha. En todos los casos hay que esperar la respuesta del ordenador, ya que no basta con pulsarla una vez.

Las teclas O y P, son las que controlan la potencia de los motores, O la reduce y P la aumenta.

El timón de cola se controla con las teclas W y R. Esto permite controlar el avión una vez en tierra, aunque también se utiliza para ayudar a realizar los giros en el aire.

Los flaps son partes de las alas de los aviones que se usan en la maniobra de aterrizaje, estos se controlan con las teclas A y F. Cuando están cerrados totalmente aparecerá una 0 y cuando estén totalmente abiertos veremos un 30.

El tren de aterrizaje se controla con la G. Hay que tener cuidado en no bajarlo en mitad del vuelo ya que esto podría acarrear nefastas consecuencias.

De nuevo insistimos en mantener las teclas pulsadas hasta que el avión responda a nuestra orden.

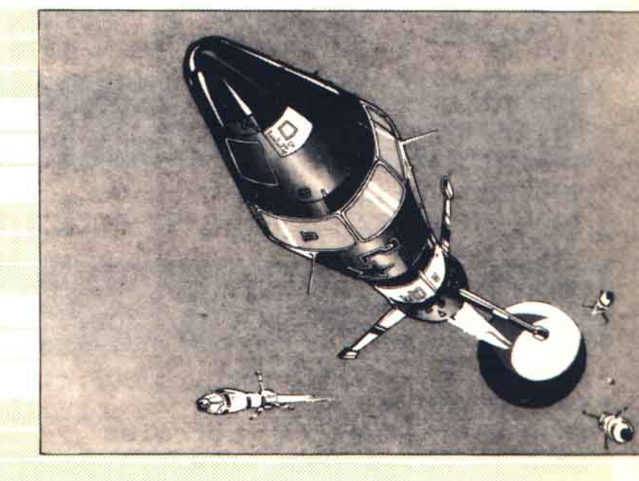
Como es normal, en todos los mapas aparece una leyenda que indica las diversas altitudes del terreno. En nuestro caso los colores indicarán la siguiente configuración: *verde,* terrenos hasta 100 metros de altitud, *rojo,* terrenos hasta 1.000 metros de altitud, y en *negro* todas aquellas zonas

de hasta 4.000 metros de altitud. Hay que poner especial atención cuando sobrevolamos estas zonas, de lo contrario te podrías estrellar.

Destaquemos las tres opciones que aparecen al ejecutar el programa. Estas son: despegar, aterrizar o la de comenzar en vuelo, con estas últimas nos evitamos la tarea de despegar, pero tendremos el problema adicional de tener que dirigirnos hacia el aeropuerto más cercano para aterrizar y repostar.

Estas dos últimas operaciones hay que hacerlas de la manera siguiente. La aproximación se ha de efectuar por el Este, es decir, en la dirección de 270 grados. La altura ha de estar por debajo de los 2.000 metros, dedicando una atención especial a la pista de aterrizaje que irá apareciendo según estemos acercándonos. Una vez hayamos tocado tierra hay que apagar los motores y quitar toda la potencia, de lo contrario provocarás un accidente.

```
100 REM********************
110 REM*                  *
120 REM*simulador de vuelo*
130 REM*                  *
140 REM********************
150 REM
160 X=RND(-TIME)
170 OPEN "GRP:" AS #1
180 REM Menu de Opciones
190 CLS
200 SCREEN 0
210 COLOR 1,15,1
220 LOCATE 3,3:PRINT "Simulador de vuelo"
230 LOCATE 3,8:PRINT "1. Despegue"
240 LOCATE 3,10:PRINT "2. Aterrizaje"
250 LOCATE 3,12:PRINT "3. En vuelo"
260 LOCATE 3,18
270 PRINT "Opción?"
280 Q$=INKEY$:IF Q$="" THEN 280
290 IF Q$>"3" OR Q$<"1" THEN 280
300 Q=VAL(Q$)
310 LOCATE 3,20
320 PRINT "Viento?"
330 A$=INKEY$:IF A$="" THEN 330
340 IF A$="s" THEN WN=1
350 SCREEN 2
360 CLS
370 ON Q GOSUB 4570,4690,4840
380 REM rutina principal
390 GOSUB 1480
400 GOSUB 760
410 GOSUB 1480
420 GOSUB 2300
430 GOTO 390
440 REM rutinas
450 IF LEN(P$)>1 THEN 480
460 P$="    "+P$
470 GOTO 620
480 IF LEN(P$)>2 THEN 510
490 P$="   "+P$
500 GOTO 620
510 IF LEN(P$)>3 THEN 540
520 P$="  "+P$
530 GOTO 620
540 IF LEN(P$)>4 THEN 620
550 P$=" "+P$
560 GOTO 620
570 IF LEN(P$)>1 THEN 600
```



```
580 P$="  "+P$
590 GOTO 620
600 IF LEN(P$)>2 THEN 620
610 P$=" "+P$
620 IF LEFT$(P$,1)=CHR$(219) OR P$=CHR$(43) THEN 680
630 IF MID$(P$,2,1)>="0" AND MID$(P$,2,1)<="9" THEN P$=RIGHT$(P$,LEN(P$)-1)
640 C=C-1
650 GOSUB 710
660 C=C+1
670 COLOR 11
680 PRESET((C-1)*8,(R-1)*8)
690 PRINT#1,P$
700 RETURN
710 PRESET((C-1)*8,(R-1)*8)
720 COLOR 1
730 PRINT#1,STRING$(LEN(P$)+2,219)
740 RETURN
750 REM Actualiza altitud, velocidad y fuel
760 FU=FU-PW*.013
770 IF FU<=0 THEN FU=0:PW=0
780 P$=STR$(INT(FU))
790 R=15:C=4:GOSUB570
800 VE=(PW*1.14-B2*44.978-VE*((FL>0)*(VE>100))*FL/150-.1*VE*
((GR=0)*(VE>100))+VE
/2*(VE<10)+VE*4)/5
810 IF VE<20 THEN 830
820 VE=VE+RND(1)*4-2
830 IF VE<0 THEN VE=0
840 P$=STR$(INT(VE))
850 R=18:C=4:GOSUB570
860 AL=AL+RO/10
870 IF AL<0 THEN AL=0
880 P$=STR$(INT(AL))
890 R=16:C=16:GOSUB 450
900 REM Comprobar accidente
910 IF AL<1 AND TS<>6 AND TS<>10 THEN 4090
920 IF AL<1 AND GR=1 THEN 4270
930 IF AL<1 AND ABS(B3-B1)>3 THEN 4270
940 IF AL<1 AND RO<-320 THEN 4090
950 IF AL<1 AND VE<1 AND (TS=6 OR TS=10) AND RO<0 AND RO>-320 THEN 4270
960 IF AL<100 AND PS=12 AND WM=2 THEN 4090
970 IF AL<1000 AND PS=6 AND WM=2 THEN 4090
980 IF AL<4000 AND PS=1 AND WM=2 THEN 4090
990 REM Velocidad de ascenso
1000 RO=(((B2*62.4-ABS(B3-B1)*22.28)*31.33)-(220*VE/150)*(AL>5)*SGN(RO)
-VE*B2*2*
(B2<-1)+RO*2)/3
1010 GOSUB 1030
1020 GOTO 1170
1030 IF AL<5 THEN 1060
1040 RO=RO+RND(1)*8-4
1050 REM Visualizar cotroles
1060 IF RO>=0 THEN 1110
1070 P$=STR$(ABS(INT(RO)))
1080 PRESET(17*8,17*8):COLOR 1:PRINT#1,STRING$(3,219)
1090 PRESET(15*8,17*8):COLOR 11:PRINT#1,"ROD";:COLOR 6:PRINT#1,CHR$(219)
1100 GOTO 1140
```

```
1110 PRESET(17*8,17*8):COLOR 1:PRINT#1,STRING$(3,219)
1120 PRESET(15*8,17*8):COLOR 11:PRINT#1,"ROC";:COLOR 1:PRINT#1,CHR$(219)
1130 P$=STR$(INT(RO))
1140 R=19:C=16:GOSUB 450
1150 RETURN
1160 REM caida
1170 IF VE>(80-FL*.66) OR AL<5 THEN 1320
1180 RO=RO-ABS(RO*RND(1)/2)-RND(1)*320*(80-VE)
1190 GOSUB 1030
1200 PLAY "acacacac"
1210 IF RND(1)>.5 THEN 1240
1220 K=69
1230 GOTO 1560
1240 IF RND(1)>.5 THEN 1270
1250 K=83
1260 GOTO 1560
1270 IF RND(1)>.5 THEN 1300
1280 K=68
1290 GOTO 1560
1300 K=88
1310 GOTO 1560
1320 IF AL<20000 OR RO<0 THEN 1350
1330 IF AL=0 THEN RO=0:GOTO 1350
1340 RO=RO*15000/AL
1350 IF VE>350 THEN 1200
1360 IF VE<140 OR AL>0 THEN 1460
1370 GOSUB 4020
1380 REM Rodando muy rapido
1390 P$="Ruedas"
1400 R=15:C=8:GOSUB 620
1410 P$="muy"
1420 R=17:C=8:GOSUB 620
1430 P$="rapido"
1440 R=19:C=8:GOSUB 620
1450 GOTO 4090
1460 RETURN
1470 REM comprobación de tecla pulsada
1480 K$=INKEY$
1490 IF K$="" THEN 2280
1500 K=ASC(K$)
1510 PLAY "o6;c"
1520 REM control por joysticks
1530 IF K<>69 AND K<>88 AND K<> 83 AND K<>68 THEN 1960
1540 IF AL<5 AND VE<(80-FL*.666) THEN 1960
1550 IF K<>88 AND AL<4 THEN 1960
1560 IF B1>6 OR B1<-5 THEN 1580
1570 R=17+B1:C=24:P$=STRING$(2,219):COLOR 15:GOSUB 620
1580 IF B2>6 OR B2<-5 THEN 1600
1590 R=17+B2:C=26:P$=STRING$(2,219):COLOR 15:GOSUB 620
1600 IF B3>6 OR B3<-5 THEN 1620
1610 R=17+B3:C=28:P$=STRING$(2,219):COLOR 15:GOSUB 620
1620 IF K<>69 THEN 1670
1630 B1=B1-1
1640 B2=B2-1
1650 B3=B3-1
1660 GOTO 1790
1670 IF K<>88 THEN 1720
1680 B1=B1+1
1690 B2=B2+1
```

```
1700 B3=B3+1
1710 GOTO 1790
1720 IF K<>83 THEN 1760
1730 B1=B1-1
1740 B3=B3+1
1750 GOTO 1790
1760 IF K<>68 THEN 1790
1770 B1=B1+1
1780 B3=B3-1
1790 IF B2<10 THEN 1840
1800 B1=-10
1810 B2=-10
1820 B3=-10
1830 REM Pirueta aerea
1840 IF ABS(B3-B1)<18 THEN 1860
1850 SWAP B1,B3
1860 IF ABS(B3-B1)<8 THEN 1900
1870 B1=B1-1
1880 B2=B2-1
1890 B3=B3-1
1900 IF B1>6 OR B1<-5 THEN 1920
1910 R=17+B1:C=24:COLOR 6:P$=CHR$(219)+CHR$(219):GOSUB 620
1920 IF B2>6 OR B2<-5 THEN 1940
1930 R=17+B2:C=26:P$=CHR$(219)+CHR$(219):COLOR 6:GOSUB 620
1940 IF B3>6 OR B3<-5 THEN 1960
1950 R=17+B3:C=28:P$=CHR$(219)+CHR$(219):COLOR 6:GOSUB 620
1960 IF K<>79 OR PW<10 THEN 1990
1970 PW=PW-10
1980 GOTO 2020
1990 IF K<>80 OR PW>210 THEN 2040
2000 IF FU=0 THEN 2040
2010 PW=PW+10
2020 P$=STR$(PW)
2030 R=21:C=4:GOSUB 570
2040 IF K<>65 OR FL<5 THEN 2070
2050 FL=FL-5
2060 GOTO 2090
2070 IF K<>70 OR FL>25 THEN 2110
2080 FL=FL+5
2090 P$=STR$(FL)
2100 R=22:C=16:GOSUB 570
2110 IF K<>82 OR RD>1 THEN 2140
2120 RD=RD+1
2130 GOTO 2160
2140 IF K<>87 OR RD<-1 THEN 2190
2150 RD=RD-1
2160 P$=" "+STR$(RD)+" "
2170 R=12:C=17:GOSUB 620
2180 REM tren de aterrizaje
2190 IF K<>71 THEN 2280
2200 IF AL<4 THEN 2280
2210 IF GR=1 THEN 2250
2220 GR=1
2230 P$="subido"
2240 GOTO 2270
2250 GR=0
2260 P$="bajado"
2270 R=23:C=8:GOSUB 620
2280 RETURN
```

```
2290 REM calcula nueva ruta
2300 HD=HD-(B3-B1)*3-RD*2*(VE>2)
2310 M2=M2-1
2320 IF AL>0 OR VE<2 THEN 2340
2330 HD=HD+RD*15
2340 IF HD>359 THEN HD=HD-360
2350 IF HD<0 THEN HD=360+HD
2360 REM comprobar rumbo
2370 P$=STR$(HD)
2380 R=12:C=9:GOSUB 570
2390 IF WN=0 THEN 2440
2400 REM calcula efecto del viento
2410 IR=COS(WD/57.296)*WS/160
2420 IC=SIN(WD/57.296)*WS/160
2430 REM calcula nueva posición
2440 PR=PR-(VE/100)*COS(HD/57.296)+IR
2450 PC=PC+(VE/100)*SIN(HD/57.296)-IC
2460 IF WM=1 THEN 2600
2470 IF PC>40 AND PC<310 AND PR>15 AND PR<100 THEN 2510
2480 PR=OI
2490 PC=OC
2500 REM mostrar nueva posición
2510 PRESET((INT(OC/10+.5)-1)*8,(INT(OI/10+.5)-1)*8)
2520 COLOR PS
2530 PRINT#1,CHR$(219)
2540 PS=POINT((INT(PC/10+.5)-1)*8,(INT(PR/10+.5)-1)*8)
2550 PRESET((INT(PC/10+.5)-1)*8,(INT(PR/10+.5)-1)*8)
2560 COLOR 15
2570 PRINT#1,CHR$(43)
2580 OI=PR:OC=PC
2590 GOTO 2700
2600 PRESET((INT(TC+.5)-1)*8,(INT(TR+.5)-1)*8)
2610 COLOR TS
2620 PRINT#1,CHR$(219)
2630 TC=TC+(VE/120)*SIN(HD/57.296)+IC*(AL<1)*VE/80+IC*(AL>0)
2640 TR=TR-(VE/120)*COS(HD/57.296)-IR*(AL<1)*VE/80-IR*(AL>0)
2650 IF TR>10 OR TR<2 OR TC>30 OR TC<4 THEN 2960
2660 TS=POINT((INT(TC+.5)-1)*8,(INT(TR+.5)-1)*8)
2670 R=INT(TR+.5):C=INT(TC+.5)
2680 COLOR 13
2690 P$=CHR$(43):GOSUB620
2700 IF WM=2 AND AL<2000 AND PC<295 AND PC>267 AND PR<55 AND
PR>41 THEN 2710 ELSE 2830
2710 IF WM=1 THEN 3010
2720 IF M2>0 THEN 3010
2730 GOSUB 3630
2740 GOSUB 3450
2750 WM=1
2760 M2=3
2770 TR=PR-35
2780 TR=TR/2
2790 TC=PC-264
2800 TS=POINT((INT(TC+.5)-1)*8,(INT(TC+.5)-1)*8)
2810 GOTO3010
2820 REM cambia el mapa
2830 IF WM=2 AND AL<2000 AND PC<85 AND PC>58 AND PR<75 AND
PR>61 THEN 2840 ELSE 3010
2840 IF WM=1 THEN 3010
```

```
2850 IF M2>0 THEN 3010
2860 GOSUB 3630
2870 GOSUB 3450
2880 WM=1
2890 M2=3
2900 TR=PR-55
2910 TR=TR/2
2920 TC=PC-55
2930 TS=POINT((INT(TC+.5)-1)*8,(INT(TR+.5)-1)*8)
2940 GOTO 3010
2950 IF M2>0 THEN 3010
2960 IF WM=2 THEN 3010
2970 GOSUB 3700
2980 WM=2
2990 M2=8
3000 PS=12
3010 IF M2<>0 OR WM<>2 THEN 3040
3020 R=7:C=7:P$=CHR$(219):COLOR 11:GOSUB 620
3030 R=5:C=28:COLOR 11:P$=CHR$(219):GOSUB 620
3040 RETURN
3050 REM limpia parte inferior de la pantalla
3060 P$=STRING$(30,219):COLOR 1:C=3
3070 FOR R=11 TO 24
3080 GOSUB 620
3090 NEXT R
3100 C=23:P$=STRING$(8,219):COLOR 15
3110 FOR R=12 TO 23
3120 GOSUB 620
3130 NEXT R
3140 R=11:C=22:COLOR 4:P$=STRING$(10,219):GOSUB 620
3150 COLOR 4:P$=CHR$(219):C=22
3160 FOR R=12 TO 24
3170 GOSUB 620
3180 NEXT R
3190 C=31
3200 FOR R=12 TO 24
3210 GOSUB 620
3220 NEXT R
3230 R=24:C=22:P$=STRING$(10,219):COLOR 4:GOSUB 620
3240 R=17:C=24:P$=STRING$(6,219):COLOR 6:GOSUB 620
3250 P$=STRING$(7,219):C=8:COLOR 15
3260 FOR R=14 TO 20
3270 GOSUB 620
3280 NEXT R
3290 REM Panel inicial de instrumentos
3300 P$="HDG":COLOR 11:R=12:C=5:GOSUB 620
3310 P$="GAS":R=14:C=4:GOSUB 620
3320 P$="VEL":R=17:C=4:GOSUB 620
3330 P$="POT":R=20:C=4:GOSUB 620
3340 P$="TREN BAJADO":R=23:C=5:GOSUB 620
3350 P$="RDR":R=12:C=13:GOSUB 620
3360 P$="ALT":R=15:C=16:GOSUB 620
3370 P$="ROC  ":R=18:C=16:GOSUB 620
3380 P$="FLP":R=21:C=16:GOSUB 620
3390 IF WN=0 THEN 3440
3400 COLOR 1
3410 P$="AIRE":R=15:C=9:GOSUB 5130
3420 P$=STR$(WS)+" KN":R=17:C=7:GOSUB 5130
```

```
3430 P$=STR$(WD)+" DEG":R=19:C=7:GOSUB 5130
3440 RETURN
3450 COLOR 1:P$=STRING$(30,219):R=1:C=3:GOSUB 620
3460 P$=CHR$(219):C=3
3470 FOR R=2 TO 11
3480 GOSUB 620
3490 NEXT R
3500 C=32
3510 FOR R=2 TO 11
3520 GOSUB 620
3530 NEXT R
3540 C=2
3550 FOR R=1 TO 24:GOSUB 620
3560 NEXT R
3570 R=4:C=7:COLOR 12:P$=STRING$(11,219):GOSUB 620
3580 R=5:C=7:COLOR 10:GOSUB 620
3590 R=6:C=7:P$=STRING$(11,219):GOSUB 620
3600 R=7:C=7:P$=STRING$(11,219):GOSUB 620
3610 R=8:C=7:COLOR 12:GOSUB 620
3620 RETURN
3630 P$=STRING$(28,219):C=4
3640 COLOR 15
3650 FOR R=2 TO 10
3660 GOSUB 620
3670 NEXT R
3680 RETURN
3690 REM mapa del terreno
3700 RESTORE
3710 FOR J=2 TO 10
3720 FOR K=4 TO 31
3730 READ CH,RP
3740 IF CH=97 THEN COLOR 1
3750 IF CH=112 THEN COLOR 12
3760 IF CH=120 THEN COLOR 6
3770 IF CH=128 OR CH=129 THEN COLOR 10
3780 P$=STRING$(RP,219)
3790 R=J:C=K:GOSUB 620
3800 K=K+RP-1
3810 NEXT K,J
3820 REM datos para el mapa del terreno
3830 DATA 97,2,120,5,112,2,120,2,112,3,120,1,97,3,120,2,112,8
3840 DATA 97,1,120,2,112,10,120,1,97,6,120,2
3850 DATA 112,5,120,1,120,2,112,10,120,7,97,2
3860 DATA 120,2,112,2,120,2,97,1,112,19
3870 DATA 120,1,97,1,112,3,129,1,112,1,120,2
3880 DATA 112,8,120,2,112,8,97,1,120,1,112,7
3890 DATA 120,1,112,3,129,1,112,2,120,8,97,5,120,3,112,6
3900 DATA 112,5,120,4,112,2,120,5,97,3,120,2,112,7
3910 DATA 112,4,120,4,112,7,120,1,97,1,120,3
3920 DATA 112,8,112,2,120,6,112,6,120,4,112,4,120,6,120,2
3930 DATA 97,2,120,1,112,5,120,2,112,8,120,4,97,4
3940 RETURN
3950 REM condiciones iniciales
3960 FU=500
3970 IF WN=0 THEN 4000
3980 WS=INT(RND(1)*30)+4
3990 WD=INT(RND(1)*360)
4000 OI=PR:OC=PC
```

```
4010 RETURN
4020 P$=STRING$(7,219):C=8
4030 COLOR 15
4040 FOR R=14 TO 20
4050 GOSUB 620
4060 NEXT R
4070 RETURN
4080 REM avión estrellado
4090 SOUND 0,0:SOUND 1,5:SOUND 2,0:SOUND 3,13
4100 SOUND 4,255:SOUND 5,15:SOUND 6,30:SOUND 7,0
4110 SOUND 8,16:SOUND 9,16:SOUND 10,16:SOUND 11,0
4120 SOUND 12,5:SOUND13,0
4130 FOR DE=1 TO 100:NEXT DE
4140 SOUND 12,56:SOUND 13,0
4150 GOSUB 4020
4160 P$="Te has":R=16:C=8:GOSUB 5130
4170 P$="estre-":R=17:C=8:GOSUB 5130
4171 P$="llado.":R=18:C=8:GOSUB 5130
4180 FOR DE=1 TO 4000:NEXT DE
4190 REM inicia las variables
4200 RO=0:PW=0:VE=0
4210 AL=0:RD=0:FL=0
4220 WN=0:WD=0:WS=0
4230 IR=0:IC=0
4240 B1=0:B2=0:B3=0
4250 GOTO 190
4260 REM compruebae el tren de aterrizaje
4270 IF GR=0 THEN 4330
4280 GOSUB 4020
4290 P$="Tren":R=16:C=9:GOSUB 5130
4300 P$="subido":R=18:C=8:GOSUB 5130
4310 GOTO 4090
4320 REM comprueba el ángulo de giro
4330 IF ABS(B3-B1)<4 THEN 4390
4340 GOSUB 4020
4350 P$="Muy":R=15:C=9:GOSUB 5130
4360 P$="incli-":R=17:C=9:GOSUB 5130
4370 P$="nado":R=19:C=9:GOSUB 5130
4380 GOTO 4090
4390 PLAY "ecf"
4400 GOSUB 4020
4410 P$="Has":R=16:C=9:GOSUB 5130
4420 P$="aterri-":R=17:C=9:GOSUB 5130
4421 P$="zado":R=18:C=9:GOSUB 5130
4430 FU=500
4440 RO=0
4450 FOR DE=1 TO 16000:NEXT DE
4460 GOSUB 4020
4470 PLAY "ce"
4480 P$="Listo":R=15:C=8:GOSUB 5130
4490 P$="para":R=16:C=8:GOSUB 5130
4500 P$="despe-":R=17:C=8:GOSUB 5130
4510 P$="gar.":R=18:C=8:GOSUB 5130
4530 FOR DE=1 TO 5000:NEXT DE
4540 GOSUB 4020
4550 GOTO 1460
4560 REM initial values
4570 HD=90
4580 PR=62
```

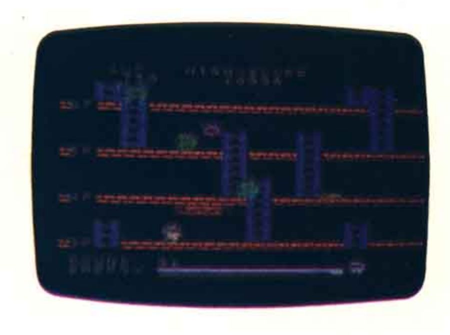
```
4590 PC=55
4600 TS=10
4610 GR=0
4620 WM=1
4630 TR=6
4640 TC=8
4650 GOSUB 3960
4660 GOSUB 3060
4670 GOSUB 3450
4680 RETURN
4690 HD=270
4700 PR=70
4710 PC=79
4720 TS=15
4730 GR=0
4740 WM=1
4750 TR=6
4760 TC=30
4770 AL=400
4780 VE=110
4790 PW=100
4800 GOSUB 3960
4810 GOSUB 3060
4820 GOSUB 3450
4830 GOTO 2020
4840 HD=INT(RND(1)*360)
4850 PR=RND(1)*50+40
4860 PC=RND(1)*160+60
4870 WM=2
4880 AL=INT(RND(1)*3000)+4000
4890 VE=INT(RND(1)*50)+100
4900 PW=130
4910 GR=1
4920 GOSUB 3960
4930 GOSUB 3060
4940 GOSUB 2220
4950 P$=STRING$(30,219)
4960 COLOR 1
4970 R=1:C=3:GOSUB 620
4980 C=3:P$=CHR$(219)
4990 FOR R=2 TO 11
5000 GOSUB 620
5010 NEXT R
5020 C=32
5030 FOR R=2 TO 11
5040 GOSUB 620
5050 NEXT R
5060 C=2
5070 FOR R=1 TO 24
5080 GOSUB 620
5090 NEXT R
5100 GOSUB 3700
5110 PS=POINT(INT(PC/10+.5)*8-1,INT(PR/10+.5)*8-1)
5120 GOTO 2020
5130 COLOR 1,15
5140 PRESET((C-1)*8,(R-1)*8)
5150 PRINT#1,P$
5160 RETURN
```

Programa: El escalador
Tipo: Juego
Distrbuidor: E.M.S.A.
Formato: Cartucho ROM

Podemos denominar este juego com el artífice instrumental para los amantes de la escalada, de tal manera, que nos hará pasar un rato trepidante e inquieto, obligándonos a desarrollar toda nuestra capacidad y habilidad para conseguir atravesar las diversas fases de que se compone el juego. Este constituye una escalada entre la vida y la muerte

Somos un solitario escalador que tiene que coronar un edificio espeluznante que está compuesto por 10 niveles, los cuales presentan clara dificultad de ascenso, porque no solamente debemos escalarlo, sino que tendremos que desarrollar nuestra pericia y tener en cuenta que existen muchos elementos más, deseosos de ver como fracasa nuestra misión.

El juego se caracteriza por la velocidad que debemos desarrollar al comenzar el ascenso, subiendo por unas escaleras, para conseguir subir de nivel.



Joysticks o teclas del cursor, elementos esenciales de nuestro **MSX** y de nuestra escalada, nos ayudarán a desarrollar nuestros reflejos y velocidad de reacción que será necesario para la consecución de nuestra misión, que aunque específicamente sea subir el último nivel del edificio, también es pasar un rato entretenido y divertido.

Entrando en la infraestructura del juego, vamos a describir las fases por las que tenemos que pasar, para un mejor entendimiento y una más clara ejecución.

Se subdivide en dos fases principales. La primera es la de guiar a nuestro héroe hasta el último piso del edificio y la segunda es, una vez llegado hasta arriba, la de subirse a una nave espacial que sobrevuela la ciudad. En el primer caso, tendremos que sortear los diversos peligros que nos acechan en la consecución de nuestro objetivo, tales como, mostruos, ratas, murciélagos, barras de hielo y diversas telarañas, que como hemos indicado, hay que esquivar. En el segundo caso y algo más relajado que en la fase anterior, nos veremos obligados a saltar a un platillo volante que está volando por encima de nuestra cabeza, este nos facilitará nuestra labor, ya que se detendrá unos momentos justo encima de nosotros. Como es lógico, el éxito de nuestra misión llegará cuando hallamos logrado subirnos a ella.

Rfiriéndonos a ls diversas subfases que posee el juego, destacaremos las 5 partes, en las cuales nos encontramos un primer momento en que nuestro escalador, alias "noso-

tros", pasa por un mal trago al tener que esquivar todos los obstáculos que se interponen en nuestro camino. Partimos con 3 vidas que tendremos que defender a capa y espada, para ello pulsando el botón de disparo podremos erigir una barrera que será infranqueable por los monstruos o saltar por encima de los obstáculos. Hay que tener en cuenta que cada vez que usemos la barrera, perderemos algo de energia, por lo que conviene no abusar de ella, puesto que si nos quedamos sin ella será imposible cumplir nuestro objeto. La tecla de disparo tiene más funciones, así, cuando deseemos saltar por encima de cualquier obstáculo, la pulsaremos y nos evitaremos problemas, el único punto negro de su utilización, es que no surte efecto si estamos subiendo por una escalera.

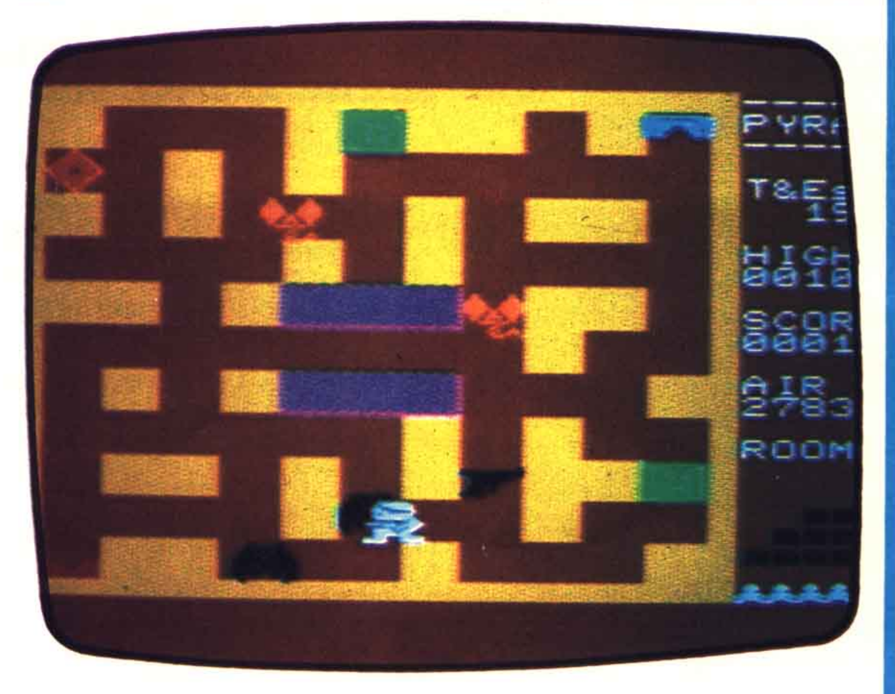
Nuestros peores enemigos son, sin lugar a dudas, los incansables e inacabables mostruos, que siempre estarán detrás de nosotros para hacer la vida imposible. Sin embargo, para eso estamos nosotros los humanos, para ayudar a nuestro pequeño a realizar su misión. Con nuestra habilidad y pericia no habrá obstáculo insalvable ni mostruo que se nos resista. Consiguiendo 10.000 puntos obtendremos un escalador adicional, no obstante, cuando hayamos perdido a nuestros escaladores, habrá finalizado el juego. Cada vez que ascendamos de nivel, nos premiarán con 250 puntos, además si llegamos al último nivel con energía, si contabilizará a nuestro favor.

La tensión y el suspense constante, nos mantedrá en vilo durante todo el juego, haciéndole intereante y entretenido. Los gráficos están bien conseguidos y la adicción irá "in crecenso".

Puntuación:
Presentación: 7
Claridad: 8
Rapidez: 8

Programa: Pyramid Ward
Tipo: Juego
Distribuidor: E.M.S.A.
Formato: Cassette

Este es el programa por excelencia, concebido para hacer agradable, entretenido, difícilmente inagualable y habilidoso, un rato de ocio, en el que nos dejemos llevar por el entusiasmo que denota el ser partícipes de un juego de estas caracteristicas.



El significado propiamente dicho del término pirámide entraña llevar innato el misterio y el peligro, desde un punto de vista menos problemático, puesto que estamos frente a un juego.

Nosotros, desde nuestros cursores serviremos de guía a un aventurero que se mueve por los laberintos que forman toda pirámide y que cada vez van siendo más complicados, de manera que nos iremos acercando más al tesoro, escondido en un cofre en el interior de la pirámide.

A lo largo de nuestro recorrido por ésta, nos encontraremos con elementos de ayuda necesarios para seguir nuestra aventura y otros que deberemos de evitar, comúnmente conocidos como nuestros enemigos. Entre los primeros, nos podremos encontrar una pistola dentro de uno de los cofres y con la cual aniquilaremos a escorpiones, murciélagos y además "monstruitos". Un segundo elemento que es un diamante que nos ayudará a entrar en el Túnel Tortuoso, el cual nos servirá para movernos instantáneamente por pantalla de un habitáculo a otro, es decir podemos pasar de un laberinto a otro.

Este túnel aparecerá de forma destellante en el centro de la pantalla, y lograremos entrar cuando se encuentre en este momento. También debemos hacer referencia a los otros elementos que son nuestros enemigos y que ya hemos mencionado antes como son los escorpiones y los murciélagos pero tenemos que destacar a la Momia Inmortal que nos aparecerá también en uno de los cofres, y a la cual no podremos aniquilar con nuestra pistola sino simple-



SOFTWARE

mente evitarla.

No obstante, el factor decisivo será conseguir el tesoro, el cual se encontrará en alguno de los laberintos que forman los tres pisos de los que se compone la pirámide. Si en nuestra larga aventura encontramos un cofre en el que haya un diamante habremos conseguido nuestro fin.

La dinámica y habilidad que debemos desarrollar no se vé referida solamente a estos elementos sino que también debemos tener en cuenta, aquellos que aunque no controlamos directamente son necesarios para conseguir nuestra meta.

Desde nuestros cursores no podremos controlarlos, pero indirectamente nuestro aventurero se verá afectado por ellos. Es decir, nos referimos a que en cada recinto hay una cantidad limitada de oxígeno, cuyo volúmen está reflejado en la parte lateral derecha de nuestra pantalla, a la vez que otros elementos, como el número de recinto en el que nos encontramos, un apartado para el record, el cual será reflejo de nuestros méritos conseguidos, y más abajo un marcador de los puntos que vayamos consiguiendo.

Tendremos a su vez una casilla con el número de aventureros que tengamos y los cuales vayamos perdiendo a lo largo del juego. Comenzaremos a jugar con cinco "exploradores" que irán desapareciendo según vayan siendo capturados o vayan agotando su aire.

Puntuación:
Presentación: 7
Claridad: 6
Rapidez: 8

Programa: Base Datos
Tipo: Aplicación
Distribuidor: ACESA
Formato: Cassette

Los forofos del orden y en general, todos aquellos que gustan de tener controlados los gastos y los ingresos (por enumerar algunas funciones de este programa), están de enhorabuena, ya que el programa en cuestión permite crear la base de datos idóneos para cada función. Es como un traje a medida, pero con la posibilidad de diseñarlo uno mismo

Imprescindible para los usuarios descritos anteriormente, sobre todo con las posibilidades que posee, ya que le ejecutarlo le permitirá introducir datos sobre un formato ya establecido o iniciar el formato más apto a sus necesidades en cualquiera de los dos casos, la respuesta del ordenador será inmediata. El único inconveniente lo encontramos a la hora de grabar o cargar datos, ya que el soporte de almacenamiento, al ser secuencial es lento.

```
FICHERO: PRUEBA          R#: 1
NOMBRE:Petrov____          REG:MSX
APELLIDOS:Ivanenko_
DIRECCION:Pza. Roja s/n.______
TF:_______       DP:_____
POBLACION:__________
PROVINCIA:__________
NOTA:█
PULSE:
      - ESC => VOLVER AL MENU
      - SELECT => OTRA ALTA
```

Iniciar el formato de la ficha es sencillo puesto que contamos con la inestimable ayuda de las teclas del cursor, que nos posicionarán en cualquier parte de la pantalla, ayudándonos en esta tarea. A continuación responderemos a la pregunta que nos hará el ordenador, sobre el nombre y la longitud del registro para preparar el fichero deseado. De esta forma se crean los registros del fichero, pero veamos como se manejan.

El menú de manejo de ficheros consta de 6 opciones, que son las siguientes: Altas, Bajas, Cosulta, Clasificación, Impresión y Fin del Trabajo.

Las primeras opciones están, como su nombre indica, para efectuar entredas o salidas de datos, aunque antes de efectuar cualquier entrada, tendremos que definirnos el formato de la ficha con la opción correspondiente. Una vez hecho esto, podremos iniciar el proceso de introducción. Las bajas, obviamente es la opción que realiza la función totalmente opuesta a la anterior, podremos borrar o o los registros que deseemos y dejar espacios para otros registros de más interés.

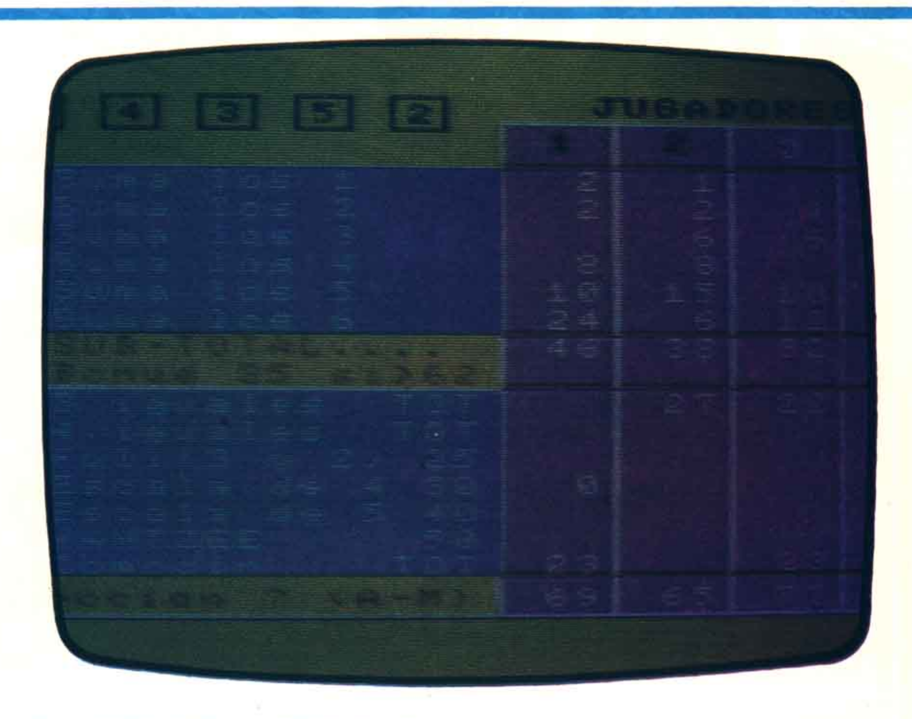
Una vez que hayamos creados el fichero correspondiente, ya sea de direcciones, de teléfonos, etc., podremos consultarlos con la opción siguiente. Esta permitirá realizar cualquier tipo de consulta con la particularidad de que podemos introducir el nombre del campo completo o solamente la primera letra, en cuyo caso, aparecerán en la pantalla todos los campos que empiecen por esa letra. Además posee otra utilidad adicional que gustará si se tienen registros distintos con algunos datos comunes.

La opción siguiente es la de clasificación, esta se podrá hacer por orden alfabético en función de un campo. Es de útil ayuda si queremos obtener una impresión de los registros ordenados alfabéticamente. Continuando con la opciones, llega-


GANE
7.000 PTAS.
todos los meses
PARTICIPANDO
EN NUESTRO
CONCURSO
A partir del próximo número MSX premiará mensualmente los programas que hagan llegar los lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.
2E1554046
BANCO
DE
ESPAÑA
1000
MIL
pesetas
2E1554046
CINCO MIL
pesetas
5000
14457150

mos a la de imprimir. No hace falta ser un genio para saber que esta permitirá obtener un listado de todos los registros que poseemos en ese momento. Por último, tenemos la opción de fin de trabajo, que permitirá hacer una grabación de todos los datos del fichero creado.

Puntuación:
Presentación: 8
Claridad: 8
Rapidez: 6



Programa: Y'Ahtzee
Tipo: Juego
Distribuidor: Dimensión New
Formato: Cassette

¿Que significa YÁHTZZE?. ¿Cómo se juega a ésto?. ¿Qué es?.

No es tan extraño y desconocido como su nombre nos puede hacer creer. Es más bien un juego antiguo y utilizado por todos aunque no hayamos jugado nunca a él, pero ¿quién no ha jugado nunca una partida de dados?.

Ya sabemos el tema de este juego que en el mercado del sofware es una de las pocas manifestaciones de cuidada creación. Las técnicas más vanzadas de la información nos hace poder tener a nuestro alcance, los juegos más habituales en reuniones, en nuestros propios ratos de ocio e incluso en campeonatos.

¿Quién después de una tarde de charla con unos amigos no ha propuesto una partida de dados?, o ¿quién más de una vez no ha deseado tener en ese momento alguien con quién jugar?, éste es y será una de las actitudes más importantes de un ordenador: "ser un compañero de juego".

El YÁHTZZE, si ya hemos jugado a los dados será fácil de comprender, no obstante introduce una serie de modalidades que hacen la partida más interesante.

Nos presenta la opción de poder jugar desde 1 hasta 4 jugadores, con sus corespodientes tres tiradas cada uno y dando paso al jugador o jugadores siguientes.

Vamos a referirnos a un jugador para simplificar y hacer más clara su comprensión. Una vez introducido al ordenador el comando de un número definido de jugadores nos aparecerá una pregunta referida al nombre del jugador (si hubiese más jugadores iría haciéndoles la pregunta a cada uno de ellos hasta tomarles a todos el nombre), con el que más tarde se referirá a nosotros durante la partida.

En la pantalla nos aparecen dos cuadros colocados verticalmente en la parte izquierda con las jugadas y los puntos que debemos obtener.

En un primer cuadro aparecen seis jugadas designadas por letras de la A a la F y que, corresponderán por orden alfabético a los números 1, 2, 3, 4, 5, 6 de los que consta cualquièr dado y de los cuales debemos conseguir la mayor cantidad posible de cada uno de ellos para así anotar el mayor número de puntos en cada una de las jugadas.

Estas seis jugadas representarán un subtotal que unidos a los puntos obtenidos con el segundo cuadro de jugadas nos permitirá erigirnos como campeones y anotar nuestro nombre en la tabla de récords que aparece al finalizar la partida.

Si nuestra suerte es buena y nuestra disposición a escoger las combinaciones adecuadas también lo es, quizás logremos superar la puntuación de 62 puntos en el primer cuadro, y seremos obsequiados con un "bonus", de 35 puntos que más tarde unidos a los puntos que cosigamos en el segundo cuadro nos permitirá inscribirnos en los récords con una mayor puntuación.

En el segundo cuadro, nos aparecerán siete jugadas cada una nombrada con una letra del abecedario de la G a la M y que tendrán asignadas diferentes puntuaciones cada una de ellas.

Nos aparecerán dos primeras opciones, G y M, que nos servirán para anotar aquellas combinaciones de tres o cuatro números iguales respectivamente, las cuales ya podemos incluir en el cuadro superior, y siendo la suma de los cinco dados la puntuación obtenida.

La I, la utilizaremos cuando consigamos un "full" es decir tres o dos números iguales. Esta opción no difícil de conseguir nos permitirá anotarnos 25 puntos. Siguiendo el orden del abecedario nos aparecen las ju-

gadas J y K, que son como las jugadas G y H anteriores pero con mayor complejidad, ya que no se trata de obtener tres o cuatro números iguales sino que tanto los tres números de la opción J como los cuatro de la opción K sigan una escala. Es un poco difícil pero no imposible, las puntuaciones serán de 30 puntos para la opción J y de 40 puntos para la opción K.

Seguirá más tarde la opción L, que tiene la mayor puntuación del juego, es decir, 50 puntos. Se trata de conseguir el juego, hacer en una palabra YAHTZEE, para ello tendremos que haber sacado en nuestras tres tiradas correspodientes cinco números iguales.

La última opción es de unas características muy necesarias para nosotros, ya que a lo largo de la partida nos veremos obligados a ir dejando una serie de jugadas, es decir, que al ir finalizando la partida nos veremos obligados a intentar obtener unas jugadas que no habíamos conseguido y aquí aparece el papel tan necesario de esta opción cuya letra es la M, y que nos servirá de "comodín" para aquella jugadas que cuando nos encontremos obligados no podremos puntuar en ninguna opción.

La puntualización que se nos dará, será la suma de los cinco números que hayamos obtenido.

Acabamos de hacer una descripción del contenido del juego y cual debe ser nuestro objetivo. Ahora nos referimos a los medios que tenemos para ello, es decir, ¿cómo serán nuestros dados?. Es lo más fácil, el cubilete tan usado y aquellos nuestros "dados de la suerte", aparecerán reflejados en nuestra pantalla por un marcador subdividido en cinco apartados que cada uno de ellos representará nuestros dados, el cual se encontrará en el margen superior izquierdo de nuestra pantalla.

El ordenador será quien tire por nosotros y aparecerá una combinación. Puede ocurrir entonces, dos opciones, una primera en la que haya un número de cifras que nos interesen, y otras que no. Las teclas de nuestro ordenador que corresponden a los números 1, 2, 3, 4, 5, nos servirán para parar o hacer que vuelvan a girar aquellos dados que nos interesen, de manera que si queremos volver a tirar los dados que ocupan las posiciones 1 y 2 pulsaremos estas teclas y aparecerán en la pantalla debajo de los marcadores dos rombos que nos indicaran los números elegidos, bastará tan solo con pulsar "return", para que estos vuelvan a girar.

El ordenador nos irá señalando a la vez la tirada en que nos encontramos, hasta agotar las tres que tenemos para cada jugada.

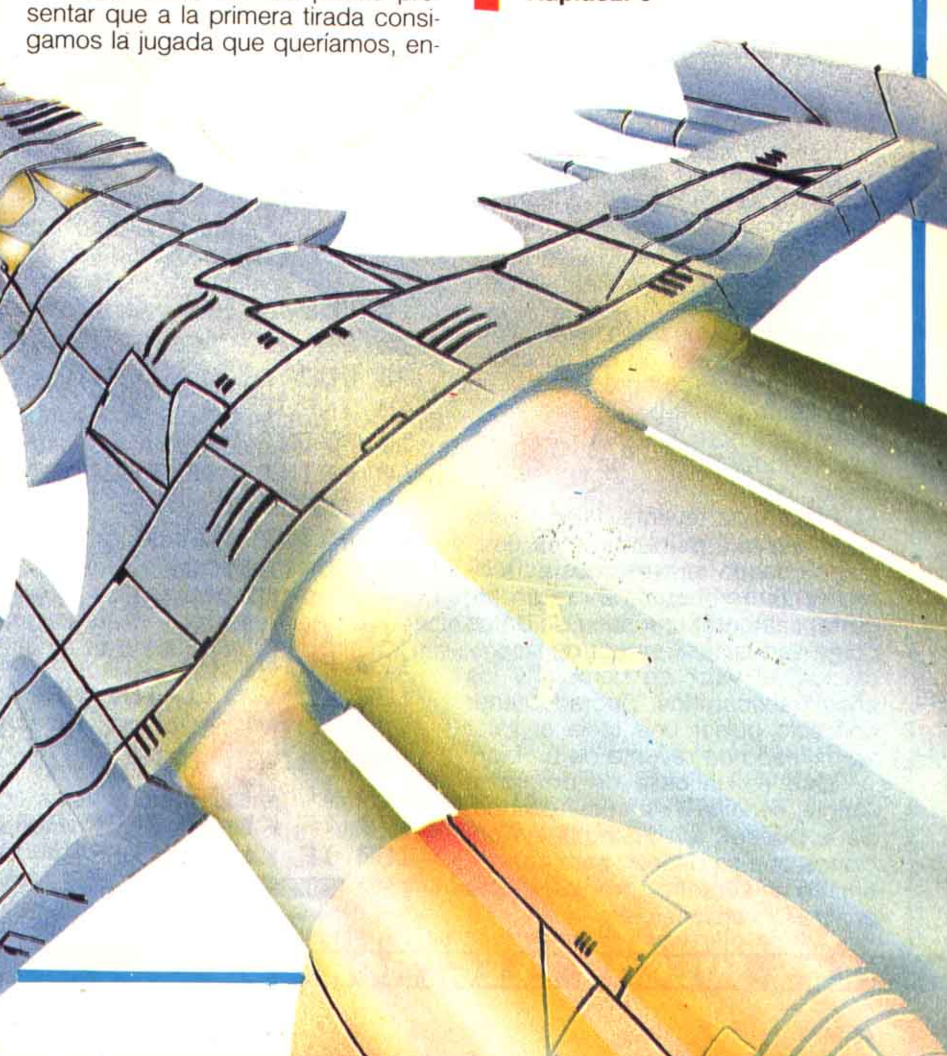
Ahora bien, se nos puede presentar que a la primera tirada consigamos la jugada que queríamos, entonces pulsaremos "return" y podremos pasar a anotar los puntos conseguidos en la jugada que hayamos conseguido.

Así sucesivamente, completaremos todas las jugadas, obteniendo una puntación que no permitirá o no ganar, **e inscribir nuestro nombre en la tabla de récords.**

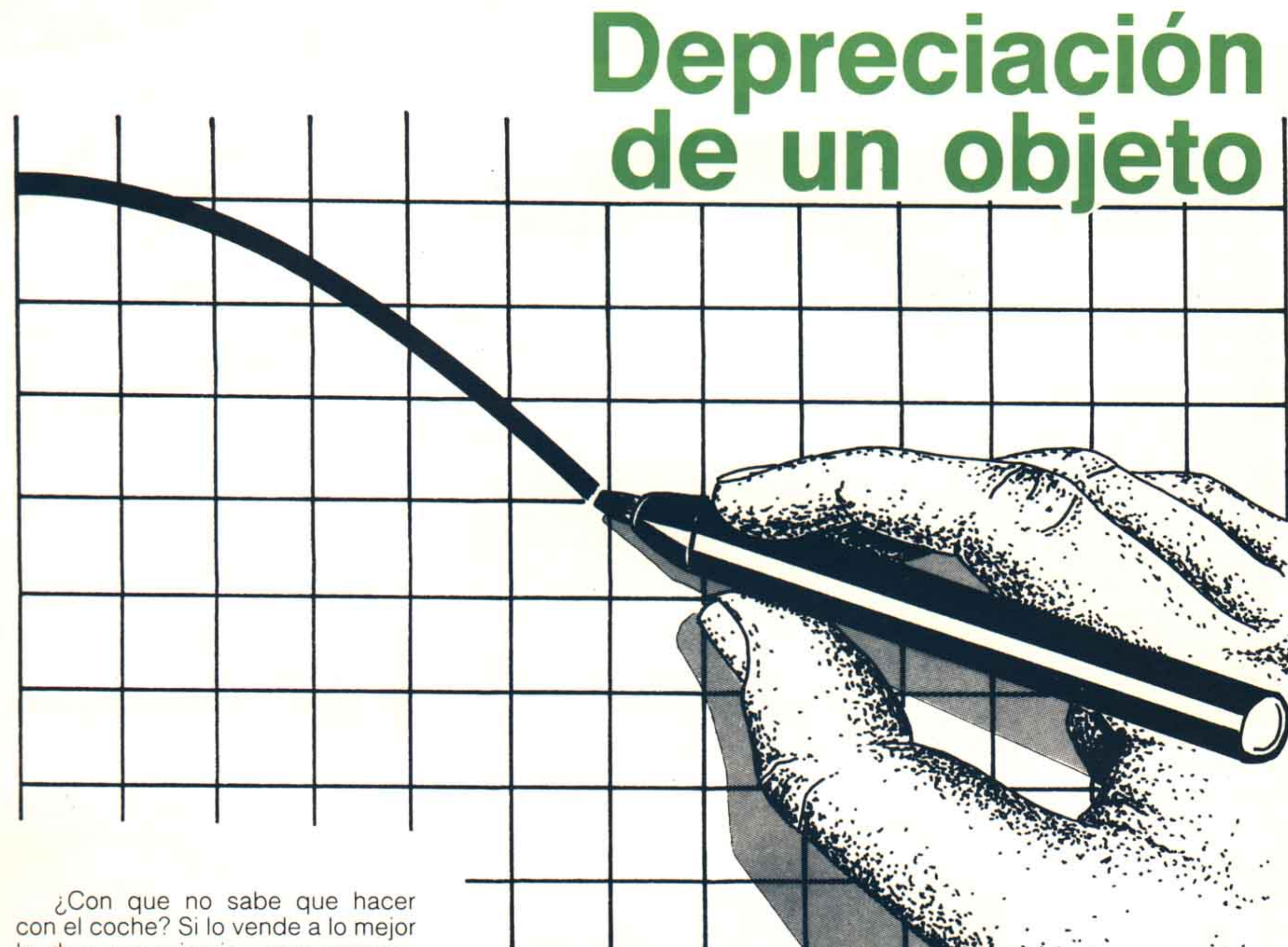
Este programa está diseñado especialmente para los "amantes de los juegos de azar", el detalle muy cuidado en él, nos permitirá pasar un rato entretenido y dinámico, con la principal característica de ese momento de incertidumbre.

Ahora sólo queda jugar y ¡suerte!.

Puntuación:
Presentacion: 8
Claridad: 9
Rapidez: 9



Depreciación de un objeto



¿Con que no sabe que hacer con el coche? Si lo vende a lo mejor le dan una miseria, pero comprar uno nuevo resulta carísimo y no están los tiempos como para ir tirando el dinero.

Este caso lo habrán vivido alguna vez, a lo mejor no era un coche lo que quería comprar dando el suyo como primer pago, pero estos apuros son normales cuando se desea vender algo de segunda mano.

El valor de reventa de un objeto suele ser un problema sólo estudiado cuando atravesamos esta situción. Sin embargo, ahora no importa, sabiendo una serie de datos como son la tasa actual de depreciación, el valor de compra y los años transcurridos, podrá obtener con solo pulsar una tecla el valor aproximado de reventa de un bien.

Este es un caso de programa simple, sencilla y de suma utilidad. Se le pueden añadir todas las opciones que se deseen para convertirlo en un auténtico programa para analizar inversiones, etc.

```
10 CLS
20 PRINT "DEPRECIACION DE UN OBJETO"
30 LOCATE 0,5:PRINT "VALOR DE COMPRA ";
40 INPUT VA
50 PRINT VA
60 PRINT "DEPRECIACION: TASA ANUAL (EN %)";
70 INPUT TD
80 PRINT TD
90 TD=TD/100
100 PRINT "Años transcurridos ";
110 INPUT NA
120 PRINT NA
130 VO=VA*(1-TD)^NA
140 PRINT "VALOR DE REVENTA ";VO
150 PRINT "OTRA VEZ?"
160 INPUT K$
170 IF K$="" THEN GOTO 170
180 IF K$="s" OR K$="S"THEN GOTO 10
190 STOP
```

DIMensionNEW A LA VANGUARDIA DEL SOFTWARE MSX

Si existe una tarea sumamente agradable para el comentarista de Software, y de la que pocas veces puede darse el lujo, es el de escribir, sobre una firma netamente nacional.
Este es el caso de DIMensionNEW, empresa fundada a mediados del año pasado y que en este corto espacio de tiempo ha logrado la consolación y el reconocimiento por parte del usuario de una calidad en sus programas a nivel internacional.

LOS PRINCIPIOS FUERON DUROS

Así fue. Inicialmente fabricando software dirigido al ordenador Spectrum, el primer problema fue conseguir entrar en un mercado saturado de programas de importación y con una mentalidad en ciertos sectores reacia a utilizar software nacional.
El primer gran paso vino dado con el contrato de comercialización de todos sus productos con el conocido SUMINISTROS VALLPARADIS, S. A. de TERRASSA, coincidiendo precisamente con las fechas en que esta firma inauguraba sus nuevas instalaciones de PASTEUR, 3.
A partir de ahí el lanzamiento fue un éxito. Sólo faltaba que un gigante de la microinformática, esta vez de Madrid, incluyera el software de DIMensionNEW en su catálogo: ABC SOFT.
El triunfo está asegurado.
Los cuatro principales objetivos quedaban cumplidos:

- Calidad del Software.
- Grabación de calidad en cinta de calidad.
- Presentación cuidada en sus mínimos detalles (DIMensionNEW fue una de las primeras marcas de Soft de Europa en presentar sus programas en estuche "tipo video").
- Comercialización seria y eficaz por parte de las firmas distribuidoras.

APARICION DEL SISTEMA MSX

Tan pronto como comenzó a hablarse a nivel internacional de este nuevo sistema el equipo directivo de DIMensionNEW tomó una decisión: Había que realizar un completo estudio técnico y de mercado para determinar la posible viabilidad de un sistema que prometía mucho, pero que parecía llegaba con retraso.
La conclusión llegó pocas semanas más tarde: El resultado del estudio daba un cierto margen de confianza al MSX en el terreno comercial, previéndose no obstante que tendría algunas dificultades de implantación en algunos países europeos.
En cuanto al estudio técnico, las dudas quedaban totalmente despejadas: Grandes equipos con un potente microprocesador (Z80 a) y muy conocido por profesionales y buenos aficionados, amplísima variedad de accesorios (Hardware) a precios asequibles. Un potente lenguaje Basic, grandes posibilidades musicales, macrolenguaje de alto nivel para realizar dibujos en alta resolución, y una larga lista de etcéteras más.

UN RETO ASUMIDO

La eficacia se demuestra a la hora de tomar decisiones importantes valorando pros y contras con rapidez. La reunión decisoria duró poco más de media hora. Las cartas estaban sobre la mesa y había que apostar fuerte.
Varias de las primeras unidades MSX que entraron en España fueron adquiridas por DIMENsionNEW y una gran parte de su equipo técnico puso manos a la obra.
Intuyendo que al principio el software que aparecería en el mercado iba a ser algo flojo en general, se propuso trabajar en dos frentes: A corto plazo, traducir al nuevo sistema los programas DIMensionNEW que con Spectrum habían sido superventas. Con ellos, y debidos a las grandes prestaciones de estos ordenadores se consiguió incluso mejorarlos. Así vieron la luz en esta versión CONTABILIDAD DOMESTICA y EL GERENTE.
A medio plazo, la previsión era desarrollar nuevos programas cuya calidad y originalidad afirmaran al software español en el mercado nacional y abrieran fronteras hacia los países europeos. Siguiendo la filosofía que anima la inspiración y aspiración de DIMensionNEW se trataba de crear un Software para aprovechar al máximo las prestaciones del ordenador.

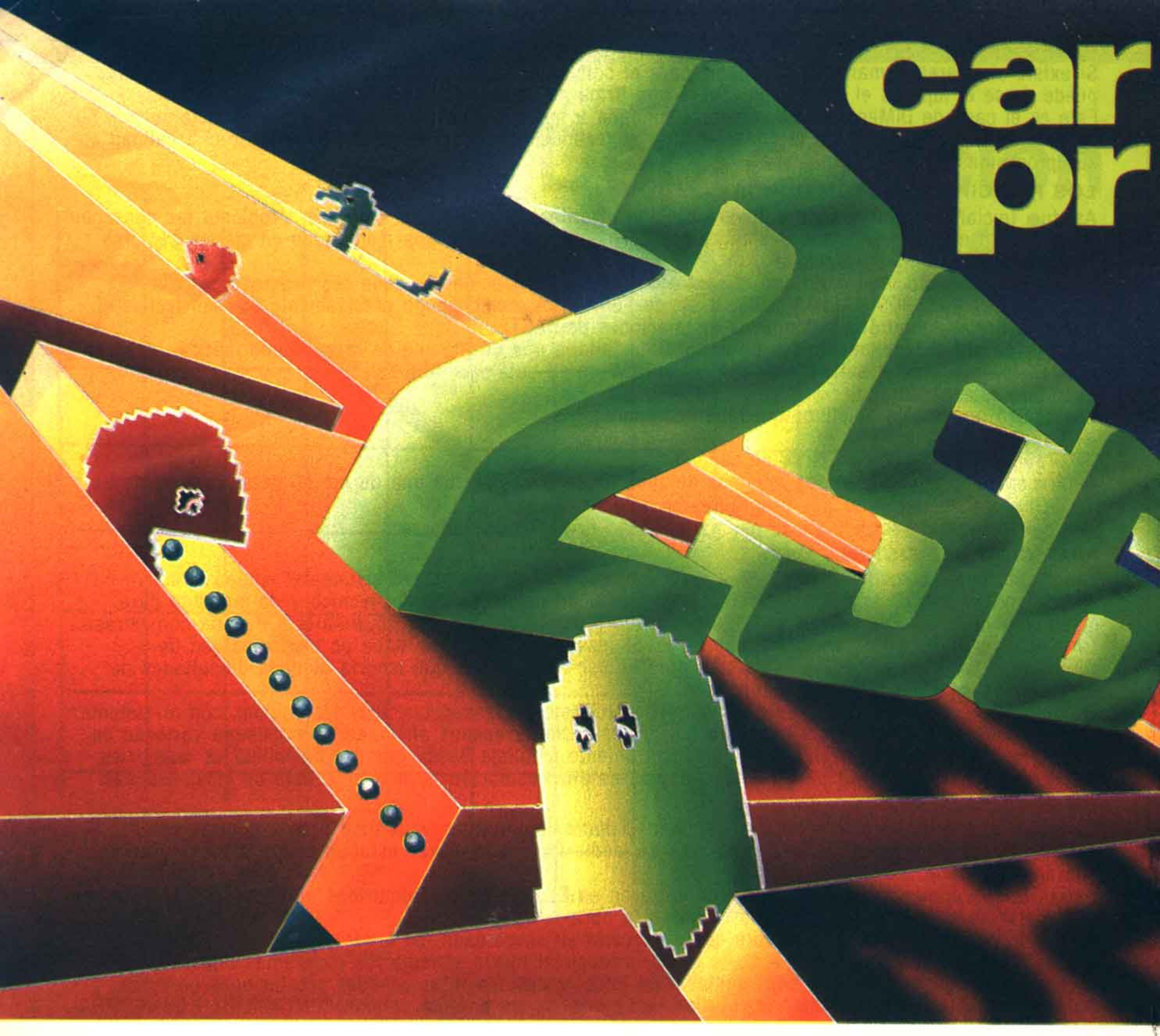
DIMensionNEW HOY

Los frutos de esta ingente tarea no se han hecho esperar y en este momento además de las firmas comerciales mencionadas anteriormente, otro gigante del software, IDEALOGIC, principalmente especializado en programas educativos y juegos intelectuales ha incorporado a su catálogo los programas DIMensionNEW. Por otra parte, sabemos existe contactos con los principales distribuidores de ordenadores MSX, habiéndose materializado estos en la producción de soft para las conocidas y prestigiosas TOSHIBA y CANON.

DIMensionNEW MAÑANA

Existia un temor por parte de los usuarios de Spectrum de que DIMensionNEW abandonara la producción de programas para este ordenador. Nada más lejos de la realidad. Nos confirman que se está preparando muy en secreto una trilogía de programas que aparecerán a finales de año. Tampoco abandonan la producción de soft para los ordenadores Dragon y Oric, e incluso se prevé la exportación al mercado francés de programas para este último.
En cuanto a MSX, DIMensionNEW está preparando una extensa variedad de programas, con soporte en cassette, cartucho y diskette que verán la luz a lo largo de este año.
Auguramos y deseamos a DIMensionNEW el éxito que se merece por su aportación al desarrollo de la microinformática en nuestro país.

R. S.

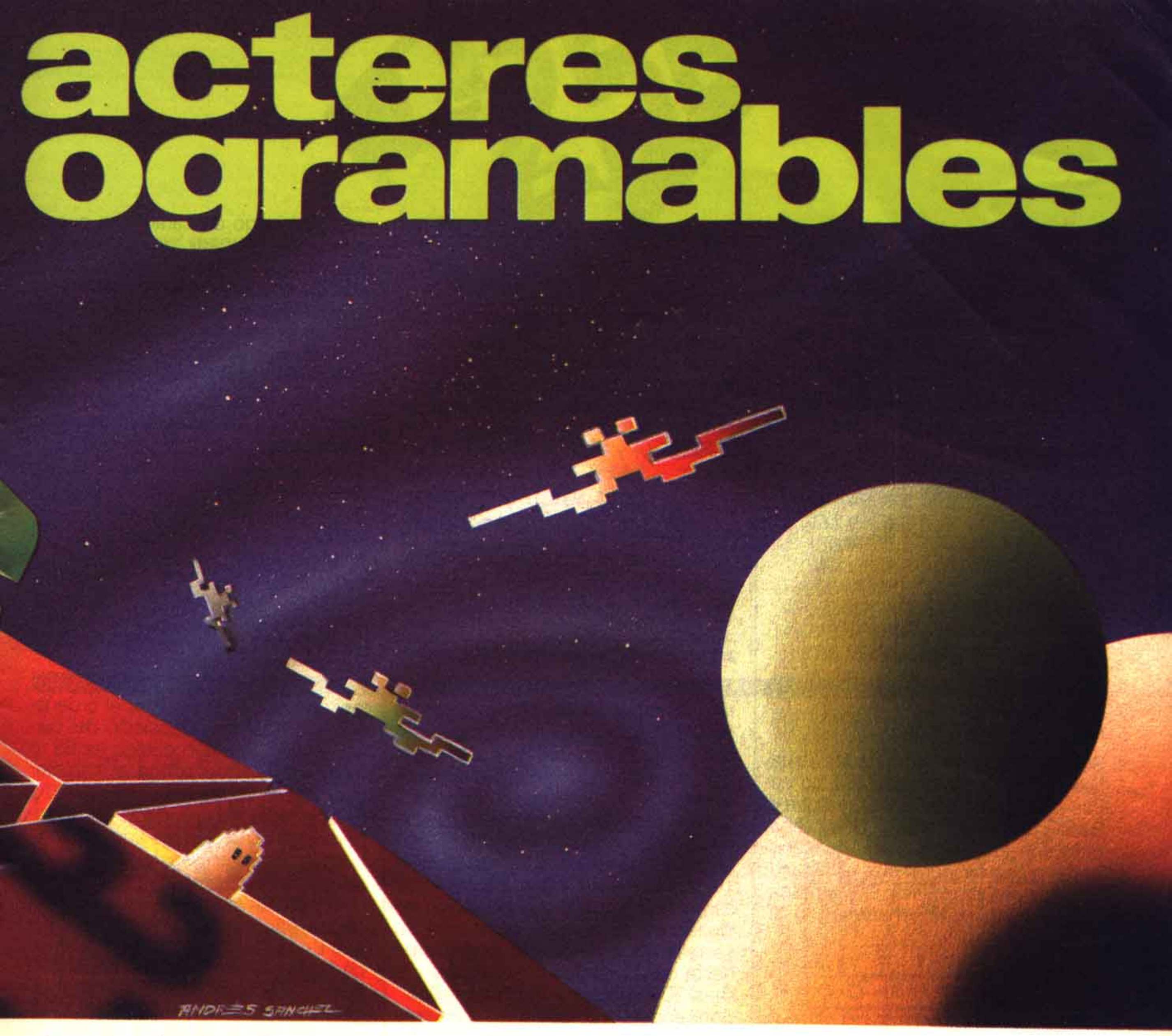


Los "caracteres programables" son algo ya muy común en la mayoría de los ordenadores domésticos. En este artículo vamos a explicar cómo se pueden crear hasta 256 caracteres gráficos en el MSX.

Seguramente, más de una vez habrás oído que otros ordenadores tienen "caracteres programables", dos palabras mágicas, gracias a las cuales puedes hacer llegar a la pantalla del ordenador desde letras góticas hasta mayúsculas acentuadas, pasando —como no— por naves extraterrestes, fantasmas o comecocos. Es verdad que en el modo de pantalla de 32 columnas × 24 líneas (SCREEN 1) esto queda subsanado por las figuras móviles, los famosos *"sprites"*, pero en el modo de textos de 40 columnas × 24 líneas (SCREEN 0) está encerrado entre las letras, gráficos y los signos de siempre. Pero hay una manera de escapar de este encierro que nos plantea el BASIC, este es utilizando la memoria de video o VIDEO RAM. Antes de explicar la forma de programar los caracteres, es indispensable poseer unas nociones sobre esta memoria y el sistema de numeración binario.

¿Qué es la memoria de video?

La memoria de video o video **RAM (VIDEO RANDOM ACCESS MEMORY)** es la parte del ordenador que está dedicada íntegramente a la pantalla. Todo lo que en un instante aparece en la pantalla, está anotado en ella y al revés; es decir, cualquier cambio realizado en la memoria de video repercutirá directamente en la pantalla. Esta memoria de video se compone de 16.384 "casilas", llamadas direcciones, numeradas del 0 al 16.383. Todas y cada una de ellas



sirven para almacenar algo, aunque no siempre lo mismo. Puede darse el caso de que una dirección tenga un significado dentro del modo de pantalla SCREEN 0, pero este valor tendrá un significado totalmente distinto dentro del resto de los modos de representación, ya sea en el modo SCREEN 1, SCREEN 2 o SCREEN 3. También puede ocurrir que encontremos una dirección que no tenga sentido alguno en cualquiera de los cuatro modos.

A una de estas direcciones se le puede asignar un número entero entre 0 y 255. Estos números son en realidad la información de lo que hay en la pantalla.
Para asignar un número o una dirección, se utiliza el mandato VPOKE, cuya sintaxis es la siguiente:

VPOKE dirección, número.

Si queremos, por ejemplo, asignar el número 121 a la dirección 13427, haremos:

VPOKE 13427, 121.

La función VPEEK se utiliza para averiguar qué número tiene asignado una dirección concreta. Por ejemplo, ahora que ya hemos introducimos 121 en la dirección 13427, si introducimos:

PRINT VPEEK (13427),

obtendremos como resultado el valor 121.
Hay que observar, que a pesar de haber introducido un valor a una dirección de la memoria video, la pantalla no ha cambiado. Una aclaración antes de continuar, el valor elegido (13427) no tiene una función específica, ni en el modo SCREEN 0 ni en SCREEN 1, pero no os preocupéis, no todos los valores son así.

¿Qué es el sistema binario?

El sistema binario es una forma más de expresar valores numéricos, o alfanuméricos. En la vida cotidiana utilizamos el sistema decimal, es decir, un sistema numérico basado en diez cifras (ver número 2 de MSX Magazine); 0, 1, 2, 3, 4, 5, 6, 7, 8, y 9. Todos los números en el sistema decimal son una combinación de estas cifras.
Pero imaginémonos que en lugar de ser una persona, somos una máquina, un ordenador, que sólo entiende si por un punto pasa o no corriente eléctrica. Las cifras que podrás llegar

a entender son sólo dos, si no pasa la corriente (0) y si pasa (1). Por lo tanto, necesitarás moverte en un sistema numérico basado en estas dos cifras, 0 y 1. El equivalente al 0 decimal sería el 0 binario, el 1 decimal sería 1 binario, pero al llegar al 2 decimal, se nos han acabado las cifras en binario, así que empezaremos a utilizar números de dos dígitos. Por lo que el 2 decimal, en binario sería 10, de esta forma podremos establecer la siguiente relación:

DECIMAL	BINARIO
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1.000

Sin embargo, los afortunados usuarios de los ordenadores MSX no necesitan complicarse la vida, ya que el BASIC de este estándar viene preparado con la función BIN$, que convierte números decimales a binario. Por ejemplo, para hallar el valor binario del número 423, haremos lo siguiente;

```
PRINT BIN$ (423)
```

el resultado será 110100111. Si ahora queremos pasar un número de binario a decimal, tendrás que poner los signos "&" y "B", y a continuación el número. Invirtiendo el ejemplo anterior, haremos;

```
PRINT &B110100111.
```

obteniendo como resultado el número 423.

Ahora, quizá comprendereis porqué los números que se pueden asignar a una dirección de la memoria video no pueden ser mayores de 255. El 255 en decimal equivale a 11111111 en binario, que es el valor máximo admitido dentro de un *byte* (8 *bits)*, siendo este la longitud de una dirección de memoria (los números de más cifras binarias no tienen sentido y a los que tengan menos de 8 dígitos, se rellenará a ceros por la izquierda hasta completar la cantidad de 8).

Los caracteres del MSX

Ahora poseemos los conocimientos suficientes como para comprender como almacena el ordenador el juego de caracteres. Pon el ordenador en el modo de pantalla SCREEN 1 (es importante, más adelante explicaremos cómo hacer caracteres en SCREEN 0) y teclea el siguiente programa;

```
10 FOR T = 0 TO 2047: A$
(VPEEK (T))
20 B$ = "0000000" : IF LEN
(A$) < 8 THEN A$ = LEFT$
(B$, 8 - LEN (A$)) + A$
30 PRINT T, A$: NEXT
```

A la izquierda de la pantalla aparecerán las direcciones de la memoria de video que contienen los caracteres (de la 0 a la 2047), y a la derecha, en sistema binario, los números asignados a cada una de las direcciones. El programa es bastante sencillo, tiene un bucle FOR-NEXT con cada una de las direcciones. La línea 10, convierte a binario los números asignados a cada una de las direcciones de la memoria, en la línea 20 se rellenan a ceros por la izquierda hasta completar los ocho dígitos y la línea 30 los imprime en la pantalla.

Sorprenderá ver cómo los primeros "1" que aparecen a la derecha forman una cara sonriente, ya que este es el primer carácter gráfico del ordenador. Lentamente aparecerán el resto del juego de caracteres que lo componen. A partir de la dirección 384, aparecerán los números, de la 520 en adelante veremos las letras mayúsculas, etcétera. La tabla de los 256 caracteres se puede ver en el manual del ordenador que posea, de cualquier manera, en el número 1 de MSX Magazine, se publicó una tabla completa de los caracteres de estos ordenadores.

En el MSX también podemos tener caracteres programables.

Por lo que hemos visto hasta el momento, comprobará que la totalidad del juego de caracteres que pueden salir en el modo de pantalla SCREEN 1 están entre las direcciones 0 y 2047, ambas inclusive. Cada carácter ocupa sólo ocho direcciones de memoria, cada una de las cuales tiene asignado un número de ocho cifras en sistema binario.

Cuando pulse la letra "A" mayúsculas, en realidad indicamos al ordenador que sitúe en la posición del cursor el carácter cuyo código es 65, o mejor dicho, que ponga el carácter que está entre las direcciones 520 y 527 de memoria de video, ¿cómo sabemos la dirección de memoria de nuestro carácter? Basta realizar una operación tan simple como el producto para averiguar la situación que ocupa dentro del juego de caracteres. En nuestro caso, esta letra cuyo código es 65 está a partir de la 520 porque 65 × 8 = 520. La letra siguiente comenzará a partir de la dirección 528. Como ya sabréis, la pantalla está dividida en 768 cuadros (24 líneas × 32 columnas) cada uno de los cuales se divide a su vez en 8 líneas × 8 columnas de pixels, que es la mínima expresión de un punto en la pantalla y de no muy fácil apreciación en solitario. Estando situado el cursor en cualquier lugar de la pantalla, al pulsar la letra correspondiente, el ordenador considerará que el carácter a imprimir tiene que poseer las características de las posiciones de la memoria a que pertenece ese carácter, en nuestro caso, las direcciones 520 a 527 que corresponden a 8 filas × 8 columnas de *pixels* (su equivalente son 8 filas de 8 dígitos binarios). Si la cifra correspondiente a un pixel es 0, ese pixel tendrá el color del fondo y si es 1, tendrá el color de la tinta. De esa forma, la "A" que a duras penas se distingue del mar de 0 que le rodea (eje-

cute de nuevo el programa anterior y comprenderá lo que queremos decir), tendrá mejor visualización con el juego de colores del fondo y la tinta.

Hagamos nuevos caracteres

Supongamos que todas las "A" que aparecen en la pantalla han de salir subrayadas. Para ello, tendremos que hacer un cambio en el patrón de esta letra, es decir, que en las direcciones de memoria que esta letra ocupa (520 a 527 inclusive) cambiaremos el código correspondiente por el de la "A" subrayada. Esto es, en la dirección 527, que es la última línea



Los sistemas de numeración se usan como cualquier otro método de transformación.

```
&B00001111 = 15
&B01111111 = 127
&B01111111 = 127
&B00000111 = 7
&B11111111 = 255
```

El siguiente programa realiza la operación anterior;

```
10 VPOKE 552, 7
20 VPOKE 553, 127
30 VPOKE 554, 127
40 VPOKE 555, 15
50 VPOKE 556, 127
60 VPOKE 557, 127
70 VPOKE 558, 7
80 VPOKE 559, 255
```

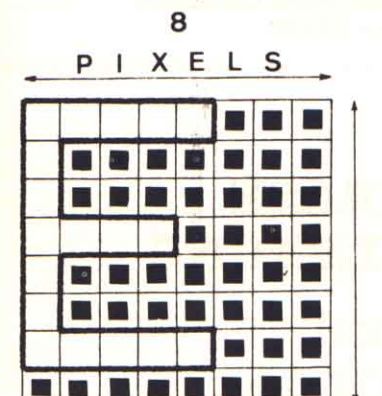


FIGURA 1

552 -	0	0	0	0	0	1	1	1
553 -	0	1	1	1	1	1	1	1
554 -	0	1	1	1	1	1	1	1
555 -	0	0	0	0	1	1	1	1
556 -	0	1	1	1	1	1	1	1
557 -	0	1	1	1	1	1	1	1
558 -	0	0	0	0	0	1	1	1
559 -	1	1	1	1	1	1	1	1

FIGURA 2

del cuadrado que forma la letra, pondremos el valor "11111111" en binario o el 255 en decimal. La operación se realiza de la forma siguiente;

```
VPOKE 527,255.
```

A partir de ahora, todas las "A" que escribamos, aparecerán subrayadas.

Por ejemplo, para invertir el color de la "E" haremos las siguientes operaciones.

En primer lugar, tendremos que crearnos el carácter (figura 1). Luego, definiremos el carácter a sustituir, por ejemplo, la "E" mayúscula, que en el código de los caracteres es el 69. Por lo tanto, las direcciones de memoria que contienen el carácter 69 son desde la 69 × 8 hasta la 69 × 8 + 7, ambas inclusive, que equivale a decir, desde la dirección 552 hasta la 559. Después habrá que pasar a binario cada una de los *bytes* (figura 2) y luego a decimal.

Los valores a introducir son los siguientes;

```
&B00000111 = 7
&B01111111 = 127
&B01111111 = 127
```

Los problemas del SCREEN 0

Para los diseñadores de ordenadores MSX, crear una pantalla con los caracteres más cerca unos de otros era relativamente fácil, pues en lugar de considerar los cuadros como 8 filas × 8 columnas de *pixels*, los toma de 8 filas × 6 columnas. De esta forma, los caracteres fundamentales, letras, números y caracteres especiales (.,/ ; =– etc.) se podían definir sin problemas, al tiempo que resulta práctico para programar. Este es el

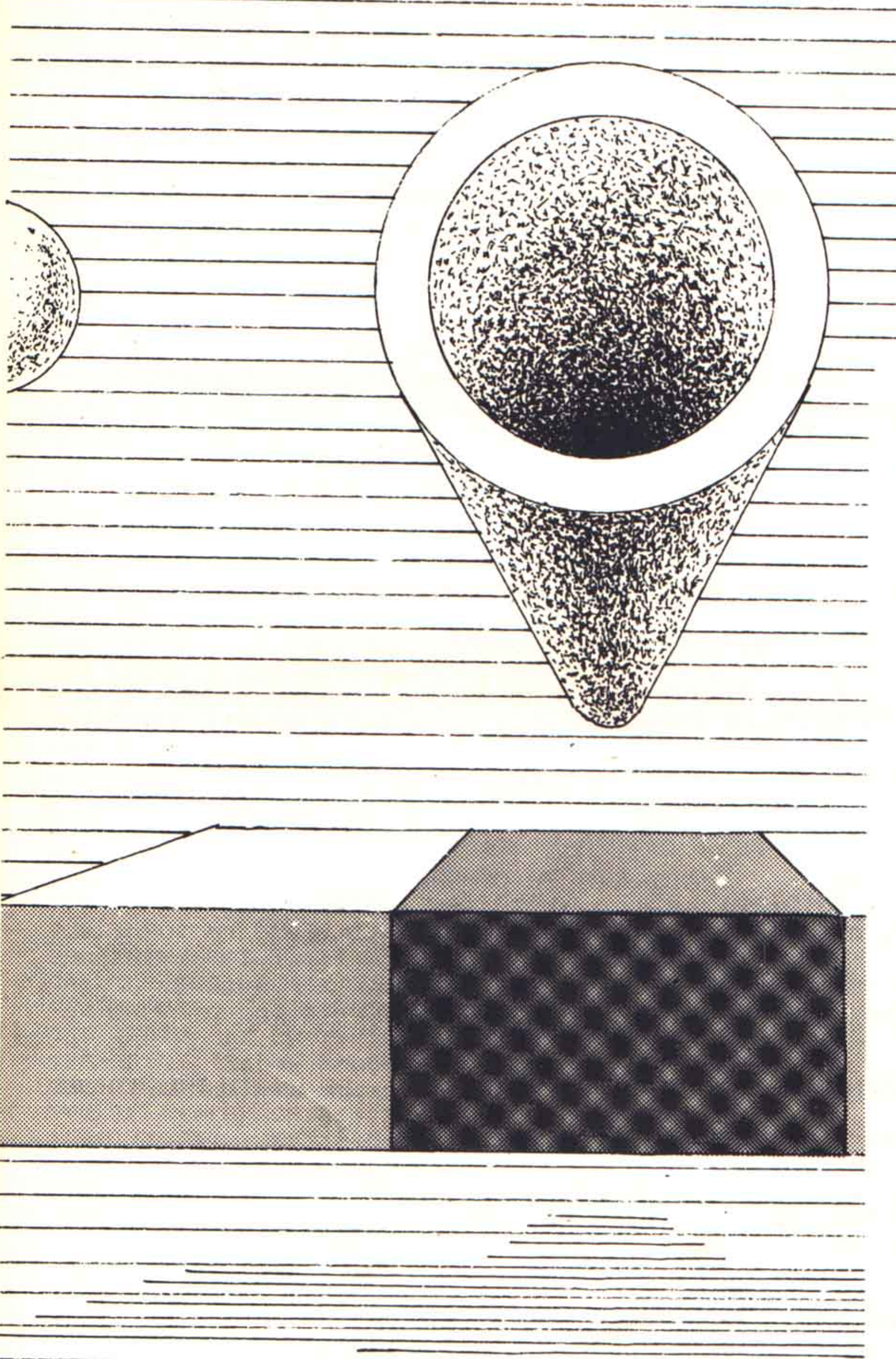
tipo de pantalla que se desarrolló con SCREEN 0, ahora bien, el problema se plantea a la hora de introducir caracteres gráficos, ya que estos sí ocupan los 8 × 8 *bits* completos, por lo que estos sólo aparecerán en parte.
Por lo tanto, hay que tener cuidado cuando definamos caracteres gráficos. En este modo de pantalla, tendremos que definirlos con las mismas dimensiones que las letras, es decir, de 8 filas por 6 columnas (hay que introducir 8 números binarios de 6 dígitos) y rellenar por la DERECHA a ceros. Resaltamos la diferencia de rellenar por este lado, por que la regla general es que se rellene por la izquierda y EN ESTE CASO ESTO NO ES POSIBLE, ya que el gráfico se cortará. De cualquier manera, es mejor hacer la prueba de este último paso y así comprobar su funcionamiento. Aconsejamos prudencia cuando empecemos a trabajar con la instrucción VPOKE. Las direcciones que almacenan los datos de los caracteres en SCREEN 0, no son del 0 al 2047, sino del 2048 a la 4095, así para sustituir la "E" por otro carácter no haremos la operación 69 × 8, sino 2048 + 69 × 8, cuyo resultado es 2600. Luego la "E" está almacenada entre las direcciones 2600 y 2607. Una última advertencia; al introducir en el ordenador los caracteres gráficos, no cambiar el modo de pantalla, puesto que perderemos todo el trabajo realizado hasta el momento y tendremos que empezar de nuevo. Permanece en un SCREEN o introduce los datos en un programa en BASIC, pues el programa no se borrará y entonces podrás volver a introducir los datos que desees.



La definición de nuevos caracteres está en función de la capacidad y conocimientos de cada usuario.

José María Cavanilles
Ana Espías

Momentos de inercia



Este programa no causará estragos, más bien servirá como recordatorio a los jóvenes usuarios, (recordatorio porque más de uno se acordará del primo de la redacción que dijo de poner este programa), además con las vacaciones por medio.

No importa, en cualquier de los dos casos, este programa a la larga será beneficioso. Estamos de acuerdo que el saber no ocupa lugar (el que se inventó la frase no tenía ordenador, de lo contrario compendería que algún que otro "k" de memoria sí ocupa).

Introducir un programa de física no tiene ningún misterio, sabido es la inestimable ayuda que da un ordenador a un estudiante y esto es un ejemplo de dicha aplicción. El programa nos pedirá una serie de datos y hallará el resultado, que podrás comprar con lo que tú realizas en el papel. Este ejemplo se puede complicar más todavía, pero eso no lo haremos nosotros. El ejemplo se divide en una serie de partes, cada una es una opción.

Estas son hallar el momento de inercia de una barra delgada, un cilindro macizo, un cilindro hueco, una chapa delgada, un disco y una esfera.

.En todos los ejemplos, aparece la fórmula de la opción elegida. Es un buen programa, del que estamos seguros sacareís buen provecho.

```
10 CLS:LOCATE 6,1:PRINT "MOMENTOS DE INERCIA"
20 LOCATE 12,4: PRINT "MENU"
30 PRINT "        1.Barra delgada                      2.cilindro macizo
          3.Cilindro hueco                  4.Chapa delgada(a*b)
     5.Disco                           6.Esfera
.Parar el prog
40 LOCATE 6,15:PRINT "Escoge una opción ";:INPUT X
50 IF X<1 OR X>7 THEN 40
60 ON X GOSUB 110,290,420,570,770,900,1070
70 CLS: PRINT "Volvemos al menu ? (s/n)"
80 INPUT X$
90 IF X$="s" OR X$="S" THEN 10
100 STOP
110 CLS : PRINT "        MOMENTO DE INERCIA                  DE UNA BARRA DELGADA
"
120 LOCATE 0,3:PRINT "DATOS:":PRINT "Masa del cuerpo (Kg):"
130 INPUT M:LOCATE 22,4: PRINT M
140 PRINT "Longitud (m):"
150 INPUT L:LOCATE 22,5: PRINT L
160 INPUT "Todo bien (s/n)";X$
170 IF X$="n" OR X$="N" THEN 110
180 PRINT:PRINT "Estoy programado para hallarlo      respecto de
        -Perpendicular al centro de gravedad -Perpendicular a un extremo"
190 INPUT" Deme el eje:(c/e)";A$
200 IF A$="e" OR A$="E" THEN 240
210 I1=(M*(L^2))/12
220 PRINT "El Momento de Inercia respecto de    perpendicular por su c.v. vale "
;I1;"Kg*m^2."
230 GOTO 260
240 I2=(M*(L^2))/3
250 PRINT "El Momento de Inercia respecto de    Perpendicular por un extremo val
e ";I2;" Kg*m^2."
260 LOCATE 0,20:PRINT "Pulse una tecla para continuar"
270 IF INKEY$="" THEN 270
280 RETURN
290 CLS:LOCATE 6,0: PRINT "MOMENTO DE INERCIA                 DE UN CILINDRO MA
CIZO"
300 PRINT "DATOS:":PRINT "Masa del cilindro (Kg)"
310 INPUT M:LOCATE 23,3:PRINT M
320 PRINT "Radio del cilindro   ";
330 INPUT R: LOCATE 23,4:PRINT R
340 INPUT "Todo bien?(s/n)";X$
350 IF X$="n" OR X$="N" THEN GOTO 290
360 PRINT "Estoy programado para hallarlo res-  pecto de su eje."
370 I=(M*(R^2))/2
380 PRINT "El momento de inercia vale          ";I;" Kg*m^2."
390 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
400 IF INKEY$="" THEN 400
410 RETURN
420 CLS: PRINT "        MOMENTO DE INERCIA                  DE UN CILINDRO HUECO
"
430 LOCATE 0,4:PRINT "DATOS:":PRINT "Masa del cilindro:"
440 INPUT M:LOCATE 22,5: PRINT M
450 LOCATE 0,6:PRINT "Radio 1"
460 INPUT R1:LOCATE 22,6: PRINT R1
470 LOCATE 0,7:PRINT "Radio 2"
480 INPUT R2:LOCATE 22,7: PRINT R2
490 INPUT "Todo bien?(s/n)";X$
500 IF X$="n" OR X$="N" THEN GOTO 420
510 PRINT "Estoy programdo para hallarlo res-   pecto de su eje."
520 I=(((R1^2)-(R2^2))*M)/2
```

```
530 PRINT "El Momento de Inercia vale          ";I;" Kg*m^2."
540 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
550 IF INKEY$="" THEN 550
560 RETURN
570 CLS: PRINT "         MOMENTO DE INCERCIA                DE UNA CHAPA DELGADA (A
*B)"
580 LOCATE 0,4:PRINT "DATOS:":PRINT"Masa de la chapa (Kg):"
590 INPUT M:LOCATE 22,5: PRINT M
600 LOCATE 0,6:PRINT "Lado A ;"
610 INPUT A:LOCATE 22,6: PRINT A
620 LOCATE 0,7:PRINT "Lado B ;"
630 INPUT B:LOCATE 22,7: PRINT B
640 INPUT "Todo bien?(s/n)";X$
650 IF X$="n" OR X$="N" THEN GOTO 570
660 PRINT "Estoy programado para hallarlo  res- pecto de: -El lado A
         -Perpendicular en su centro"
670 INPUT "Deme el eje: (A/c)";A$
680 IF A$="c" ORA$="C" THEN GOTO 720
690 I1=(M*(B^2))/2
700 PRINT "El Momento de Inercia respecto del   lado A vale ";I1;" Kg*m^2"
710 GOTO 740
720 I2=(M*(A^2)+(B^2))/12
730 PRINT "El momento de Inercia respecto de la perpendicular por centro vale ";
I2;" Kg*m^2."
740 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
750 IF INKEY$="" THEN 750
760 RETURN
770 CLS : PRINT "        MOMENTO DE INERCIA                  DE UN DISCO"
780 PRINT:PRINT "DATOS:":PRINT "Masa del disco(Kg):"
790 INPUT M:LOCATE 22,4: PRINT M
800 PRINT "Radio; "
810 INPUT R:LOCATE 22,5: PRINT R
820 INPUT "Todo bien?(s/n)";X$
830 IF X$="n" OR X$="N" THEN GOTO 770
840 PRINT "Estoy programado para hallarlo res-  pecto de su diámetro."
850 I=(M*(R^2))/4
860 PRINT "El Momento de inercia vale          ";I;" Kg*m^2."
870 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
880 IF INKEY$="" THEN 880
890 RETURN
900 CLS: PRINT "        MOMENTO DE INCERCIA                DE UNA ESFERA"
910 PRINT:PRINT "DATOS:":PRINT"Masa de la esfera:"
920 INPUT M:LOCATE 22,4: PRINT M
930 PRINT "Radio; "
940 INPUT R:LOCATE 22,5: PRINT R
950 INPUT "Todo bien?(s/n)";X$
960 IF X$="n" OR X$="N" THEN GOTO 900
970 PRINT "Estoy programado para hallarlo res-  pecto a su diametro."
980 INPUT"Esfera maciza o hueca?(m/h)";A$
990 IF A$="H" OR A$="h" THEN GOTO 1020
1000 I1=2*(M*(R^2))/5
1010 PRINT "El Momento de inercia de esfera ma-  ciza vale";I1;" Kg*m^2."
1020 I2=2*(M*(R^2))/3
1030 PRINT "El Momento de inercia de esfera huecavale";I2;" Kg*m^2."
1040 LOCATE 0,20:PRINT "Pulsa una tecla para continuar"
1050 IF INKEY$="" THEN 1050
1060 RETURN
1070 CLS: PRINT "       PROGRAMA PARADO"
```

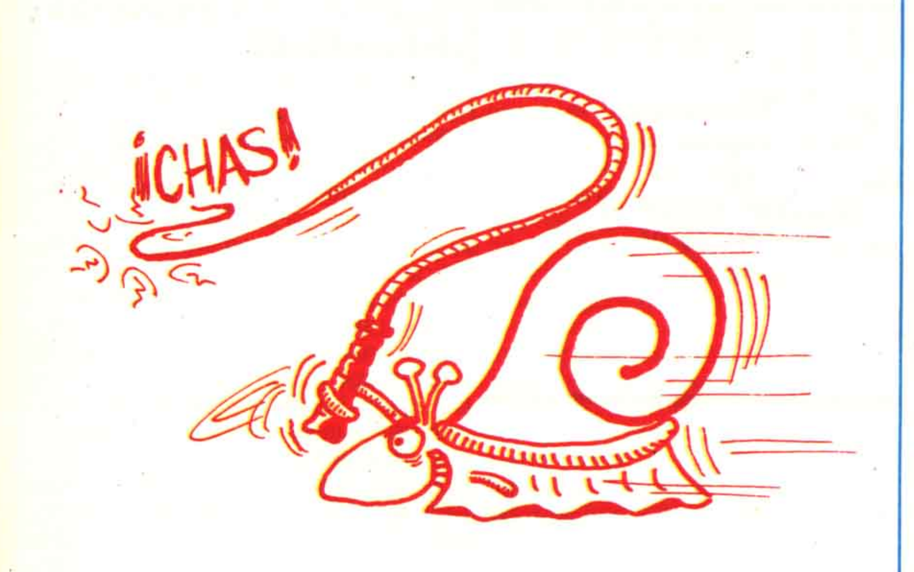
Visión panorámica de los microprocesadores más comunes



Los microprocesadores son el auténtico motor de los ordenadores. Comúnmente llamado chip, cucaracha y nombres más extraños, siempre se ha planteado la duda de cúal es el mejor para un ordenador determinado. El presente artículo, no pretende ser un curso de iniciación rápida a los microprocesadores, sino más bien, una pequeña comparación entre los chips MPU más comúnmente conocidos.

Para responder a la pregunta ¿Qué es un microprocesador? con detalle, serían necesarios varios libros, dada la complejidad y la profusión de tipos actualmente existentes. Sin embargo, la mayoría de los microprocesadores actuales, salvo alguno muy moderno, parten del mismo concepto.

La idea se le ocurrió a Von Neumann, quién sentó las bases de pro-



grama y máquina secuencial programable. Se trata de poder disponer de una máquina, que sin "grandes cambios" se pudiera utilizar para resolver tareas de cálculos diferentes. Para poder materializar esta idea es necesario disponer de una máquina que se pueda programar y por otra parte tener un programa, sucesión de órdenes o instrucciones que resuelvan el problema planteado y puede ser interpretado por nuestra máquina programable.

Que es un microprocesador

Por tanto se desprende de esta breve introducción, que un microprocesador es básicamente una máquina o autómata que es capaz de ejecutar una secuencia de instrucciones, que previamente le ha sido dada o introducida. Esta idea ha sido materializada de diferentes maneras con el paso del tiempo, según la tecnología disponible en cada momento. Se empezó con relés, pasando a válvulas y posteriormente a transistores, circuitos integrados SSI, MSI, LSI hasta llegar a la actualidad a los VLSI y dentro de poco tendremos los **VHSIC** *(Very High Sacale Integrated Circuit).*

La configuración interna que posee un ordenador o microprocesador se denomina arquitectura. Las piezas con las que está construido un microprocesador, a nivel conceptual, varían según la máquina que se trate, como veremos posteriormente al comparar los microprocesadores más comunes. Sin embargo, al partir todas del mismo concepto, tienen partes comunes. En primer lugar, puede observarse que disponen de una unidad aritmético-lógica, que es la encargada de efectuar las operaciones del programa, con los operandos que hemos especificado en los mismos. Esta unidad es la que por ejemplo, suma, resta, compara y realiza las operaciones lógicas AND, OR, NOT, etc. Como curiosidad hay que señalar, que generalmente los microprocesadores no suelen disponer de multiplicación o división, salvo los más grandes y modernos, que sí las incorporan.

También disponen los microprocesadores de una unidad de control de proceso, que es la que se encarga de buscar la primera instrucción del programa y, a partir de ésta, busca la siguiente al terminar la ejecución de la instrucción en curso. Las instrucciones que "toman" datos de la memoria o "dejan" datos en la misma, son ejecutadas por esta unidad, así como todas aquellas que tengan relación con el "control" del programa, tales como los saltos.

Estas dos unidades son las que suelen englobarse en lo que suele denominar *chip* microprocesador o **MPU** *(Micro-Procesador Unit).* La denominación **CPU** para estos *chips*, a juicio del que escribe estas líneas, es por tanto imprecisa al hacer referencia solo a una parte de lo que contienen.

La tercera unidad que constituyen los ordenadores básicos es la memoria o lugar donde reside el programa o secuencia de instrucciones que nosotros deseamos ejecutar. No nos vamos a extender con las memorias, pero vamos a aprovechar para resaltar el hecho de que como mínimo la **MPU** debe hacer un acceso al ejecutar una instrucción, siendo este acceso o lectura de la memoria precisamente para enterarse de la instrucción que debe ejecutar. Si por ejemplo se trata de una instrucción de suma de dos números, deberá efectuar como mínimo dos accesos más, para buscar ambos números.

En este punto puede introducirse el concepto de longitud de palabra de microprocesador. Si los números que debe sumar nuestra máquina son muy largos, deberá efectuar más accesos a memoria hasta sacar de esta, todas las "cifras" que componen estos números.

Eléctricamente se pueden representar fácilmente dos estados diferentes, presencia de electricidad y ausencia de ella. Podemos asociarlos con un "1" o un "0" lógicos respectivamente. Si a la vez deseamos representar más estados, podemos agruparlos en grupos de 4 por ejemplo. De esta manera se podrán reprentar 16 estados diferentes (2 ↑ 4). A cada unidad de este grupo se le denomina *bit (binary digit),* pudiéndo adoptar por tantos dos valores. Esos *bits* se van agrupando en potencias de dos, denominándose las agrupaciones de 4 *bits* (2 ↑ 2) "nible", 8 bits (2 3) *"byte"* estando menos generalizados, por depender del tipo de microprocesador los de 16 *bits* (2 ↑ 4) *"word"* y para los 32 bits (2 ↑ 5) *"long word"*. Resulta claro que a mayor longitud de bits, más posibilidades o estados se pueden representar, siendo por tanto más potentes las máquinas de mayor longitud de palabra y también más cosas. La longitud de la palabra, es por tanto, la unidad de infor-

mación de los microprocesadores, pudiendo manejar en algunos casos (por ejemplo, el 68020) subunidades o superunidades. En nuestro caso, los microprocesadores de los ordenadores domésticos suelen emplear "todavía" **MPÚs** de 8 *bits*. Esto ha sido posible en parte al formato de los chips de memoria, de los cuales una gran variedad están organizadas en 8 bits. Recientemente se ha introducido no obstante un *chip* de memoria organizado en 16 *bits*.
Si se trata por tanto de un microprocesador de 8 *bits* y pudiésemos abrirlo, veríamos paquetes de 8 *bits* en paralelo circulando en su interior.

Su funcionamiento

También serán necesarios 8 *bits* conductores para comunicar la **MPU** con la memoria, es decir, el bus de 8 *bits*.
Para realizar las sumas, restas, y demás operaciones se pueden guardar los datos en el interior de la **MPU**. De esta forma si nuestro programa tiene que sumar varias veces el mismo número, en lugar de ir a buscarlo sucesivamente a la memoria, se puede tener guardado este número en un registro interno de la **MPU**, siendo entonces el proceso de suma de nuestro programa más rápido. Para tener una idea de cuanto más rápido, puede afirmarse que como mínimo es tres o cuatro veces más veloz.
Ahora estamos en codiciones de introducir un nuevo concepto, el de reloj en un microprocesador: la realización de "salto si es cero" en el caso

Son los elementos indispensables en cualquier tipo de ordenador, desde el personal hasta el "mainframe".

una instrucción, es decir, su ejecución dura cierto tiempo o pasos de reloj. Estos "pasos" están perfectamente determinados para cada instrucción y para cada condición de la **MPU**. Con esto quiero distiguir, por ejemplo, la duración de la introducción de "salto si es cero" en el caso de que la condición sea verdadera o falsa. Si es verdadera, esa instrucción dura unos pasos, mientras que si no lo es, dura otros pasos. Este número de pasos son perfectamente conocidos. Si el reloj da un millón de pasos por segundo, cada paso durará un micro segundo, siendo la frecuencia su inverso, un Megahercio (Mhz).

Entonces si la instrucción de "salto si es cero" dura 5 pasos y 7 si no lo es, nuestro microprocesador tardará en ejecutarla 5 ó 7 microsegundos. Es evidente que cuanto más rápido vaya el reloj, es decir, cuantos más pasos por segundo dé, menos tardarán las instrucciones en ejecutarse. Este concepto puede usarse para comparar las potencias de un microprocesador, viendo cuantos mllones de instrucciones por segundo (**MPU**) es capaz de ejecutar, ya que lógicamente la tecnología con la que están construidos los *chips* tienen un techo máximo de frecuencia de reloj, o lo que es lo mismo, una duración mínima para cada paso.
Ahora vamos a comparar algunos de los microprocesadores actuales más comunes. El más popular es el **Z-80**, seguido del 6502 y los menores difundidos en ordenadores personales 6800 y 8085.

Algunos procesadores del mercado

Empecemos por describir los microprocesadores. El 6502 tiene tres registros de 8 *bits*, que son el Acumulador, el Indice X y el Indice Y además del registro de *status/flags* y dos registros de 16 *bits*, el contador de programa y el puntero de pila o de *stack*. Este último tiene la particularidad de tener el *byte* más sigificativo siempre a 01, lo que quiere decir en las posiciones 0100 y 01FF tienen que haber RAM. Con relación al juego de instrucciones, hay que señalar que este no es lineal, teniendo unas instrucciones y unos modos de direccionamiento que no poseen en su totalidad los demás instructores. Además el registro X está algo más favorecido por el juego de instrucciones que el registro Y. En total el **6502** tiene 13 modos de direccionamiento. En lo que respecta a la capacidad de direccionamiento, es la estándar de casi todos los 8 *bits* tradicionales, 64 *Kbytes*. Hay que señalar que los dispositivos de entrada/salida se manejan como memoria, robando espacio vital en esta. La velocidad de reloj típica del **6502** es de 1-2 Mhz, aunque hay versiones más veloces. Los vectores de interrupción y de reset se encuentran en las posiciones altas de la memoria, (FFFA – FFFF) lo que obliga a tener la ROM en estas posiciones. El **6502** puede considerarse como uno de los pioneros, existiendo **MPÚs** más modernas derivadas de ella. Como ejemplo, cabe citar *chip* 65C02, que es una versión **CMOS** de la **6502**, pero con importantes mejoras, tales como un juego de instruciones ampliado, conservando la compatibilidad con la "antigua" **6502**. Muy similar es el **6800**, con dos registros acumuladorers A y B de 8 *bits*, registro de status, registro índice de 16 *bits* y contador de programa, además de un puntero de *stack* también de 16 *bits*. Posee siete modos de direcionamiento y un juego de instrucciones bastante completo, incluyendo instruciones de *test* de *bits* y complementación, a "1" y a "2". La versión más rápida es de 2 Mhz, incorporando además la posibilidad de conexión de DMA al bus. Brevemente diremos que el DMA es un disposi-

tivo que sin intervención directa de la MPU mueve bloques de datos. con ello las transferencias de datos de memoria a memoria o de memoria a dispositivos de entrada/salida son mucho más rápidas. Dentro de la misma familia nos encontramos con otra MPU, la 6809, mucho más moderna que las anteriores, además de más potente, ya que incorpora dos modos de funcionamiento. Por un lado, el modo usuario y por otro el modo *hardware o Supervisor*, con su respectivos punteros de *stack*. Sobre el **6800** incorpora un registro de páginas de 8 *bits* como uno de 16 *bits*, D. El *stack* de ambos puede ubicarse donde se quiera en el mapa de memoria, siendo más flexible que el **6502**. Los vectores de Reset e interrupción, están en las posiciones más altas de la memoria, debiendo por tanto estar la ROM en estas posiciones, igual que en el **6502**. El **6809** incorpora además la posibilidad de compartir sus buses con otro CPUs iguales o DMA. En cuanto a su juego de instrucciones, posee 13 modos de direccionamiento, además de ser más potente que el del 6800, incorporando la operación del producto sin signo, que no tienen otros CPU's de 8 *bits* tradicionales. También dispone de un modo de escritura del tipo *"early write"* que facilita el uso de memorias dinámicas.

Otra rama de microprocesadores tiene al **8080** en común. De esta CPU se han derivado la **8085** y **Z-80**. Se pueden establecer por tanto una clasificación de **MPUs**, siendo la **6502, 6800** y **8080** de la primera generación y la **65C02, 6809** y **Z-80** de la segunda generación. A caballo entre

Hay múltiples MPU's en el mercado, cada uno con su característica que le define, pero básicamente son idénticos.

TERMINOLOGIA

Como todo artículo escrito a alto nivel, puede ser difícil comprender algunas expresiones o terminologías. Por esta razón y sabiendo que nuestros lectores no tienen por que saber lo que algunos términos significan, se ha elaborado la siguiente lista con los posibles términos que pueden plantear algún problema. Empecemos definiendo lo que es un vector de interrupción. Este es un dirección de memoria, clave o bits que indican la procedencia de la interrupción. Por ejemplo, si el vector 1 está asignado a la impresorta y el 2 âl diskette, al recibir la MPU el vector 1, pasará a ejecutar el programa de atención a la impresora. Las interrupciones pueden ser de distintas clases y de tipos diversos, en su momento las explicaremos con detalle. Pasemos ahora a definir lo que es un secuenciador programable. Esto es un dispositivo que genera una serie de señales, eléctricas, que permiten el funcionamiento interno de la MPU y otros dispotivos, a nivel de intercambio de señales e información entre los registros internos, buses, rutas de datos del chip en cuestión, en función de la instrucción que se esté ejecutando.

Ahora pasemos a explicar una serie de términos que la mayoría de los lectores conocerán. Se trata de los circuitos SSI, MSI, LSI,VLSI Y VHSI. Todas estas iniciales hacen referencia a circuitos integrados diseñados de un forma particular. Los SSI (Single Scale Integration) son circuitos de escala sencilla. Por ejemplo, los chips de puertas lógicas.

MSI, (Medium Scale Integration), integración a media escala, que puede ser un decodificador (circuitos diseñados para acceder a direcciones de la memoria).

LSI (Large Scale Integration), integración a muy gran escala, como pueden ser los controladores de memorias dinámicas, memorias EPROM y RAM estáticas hasta 2K, etc.

VISI, (Very Large Scale Integration), integración a muy gran escala. Aquí se engloban los MPU's, periféricos, memorias dinámicas, etc.

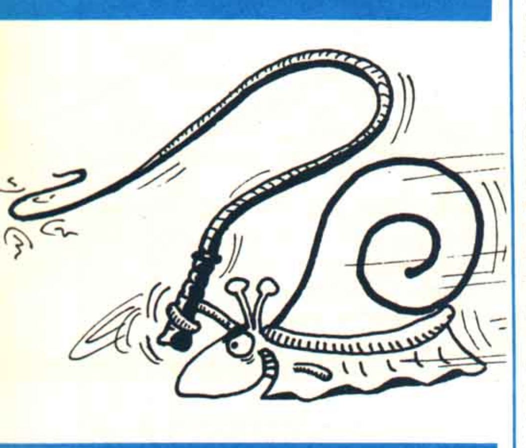
Por último, tenemos los circuitos diseñados especialmente para la tecnología del ejercito. Se trata de los VHSI, (Very High Scale Integration), integración a muy alta escala. En este caso los circuitos son ultrarápidos, que seguramente tendrán pastillas de silício en tres dimensiones.

Actualmente se emplean en la electrónica de los misiles, guías de tiro y en aviación.

Otros términos que a lo mejor son familiares aunque normalmente no se saben utilizar son los acumuladores y los registros índice. Los primeros son los registros en que se realizan las operaciones aritméticas y lógicas en los microprocesadores de 8 bits. En los de 16 y 32 bits, los avanzados y más modernos, tienen la posibilidad de realizar las anteriores operaciones en todos sus registros. En lo que se refiere a los registros índice, estos son los que permiten ser empleados de propósito general y que además puede ser utilizado como registro de exploración de posiciones de memoria. En él, lo que se guarda son índices que se van a sumar o restar a la dirección del operando.

Aquí finaliza nuestro pequeño glosario de términos interesantes y desconocidos para algunos, aunque para otros no lo sean. No dude en escribirnos si encontrara algún problema con éste o cualquier otro artículo.

ambos se encuentran el **8085** más potente al **8080** que al **Z-80**. Por ello, solo se va a tener en cuenta al **8085** y no al **8080**. El primero de estos dos, dispone seis registros de propósito general de 8 *bits*, el acumulador de 8 *bis* y contador de programa y un puntero de *stack* de 16 *bits*.
Como particularidad está el hecho de poder emplear los registros de propósito general, B, C, D, E, H, L, como registros de 16 *bits*, para efectuar únicamente direccionamientos y sumas, aparte de poderlos incrementar o decrementar. Esta MPU permite también la cesión del bus a dispositivos DMA u otros MPU's contando además con una línea de salida y otra de entrada de datos serie. A nivel de interrupciones esta MPU tiene



5 entradas de interupciones, de diferente prioridad, a diferencia de los de más MPU's descritas que sólo disponen de 2 entradas. Otra particularidad que también incorpora el **Z-80**, es la división en dos mapas de memoria y de entrada/salida. El primero es de 16 *bits*, con capacidad de direcionamiento de 64 Kbytes y el segundo permite la conexión de hasta 256 periféricos. Esto ofrece la ventaja de contar con 64 Kbytes de RAM/ROM, incorporando además cualquier periférico paralelo, como interfaces paralelo, serie, controladores de discos, de pantallas, de teclado, generadores de audio programable, etc. En las **MPÚs** descritas anteriormente, cada periférico robaba espacio de memoria, mientras que para ampliarla, se debe recurrir a

Los hay de 8 y 16 bits, todo depende de las necesidades del usuario, pequeños ordenadores de 8 bits y equipos más potentes con procesadores de 16 bits.

la técnica de la conmutación de bancos de memoria. La **MPU 8085**, cuenta con una característica única a nivel de bus, están multiplexados el bus de datos (D0-D7) con el de direcciones (A0-A7) por lo que la pastilla dispone para ambos de 8 pines únicamente (AD0-AD7, A8-A15). Junto con el pin ALE se puede distinguir cuando hay datos o direcciones por el bus. Esta técnica también se emplea en MPÚs de 16 bits. El juego de instrucciones es amplio, disponiendo de sólo cuatro modos de direccionamiento. Mediante dos instrucciones es posible transmitir o recibir *bit a bit* por el spot serie que incorpora, pudiendo además enmascarar las diferentes interrupciones según las necesidades del usuario. Hay que resaltar que estos **MPÚs,** incluida la **Z-80**, dividen por dos la frecuencia del reloj, por tanto si la frecuencia para el **8085** es de 6 Mhz, operará internamente a 3 Mhz.

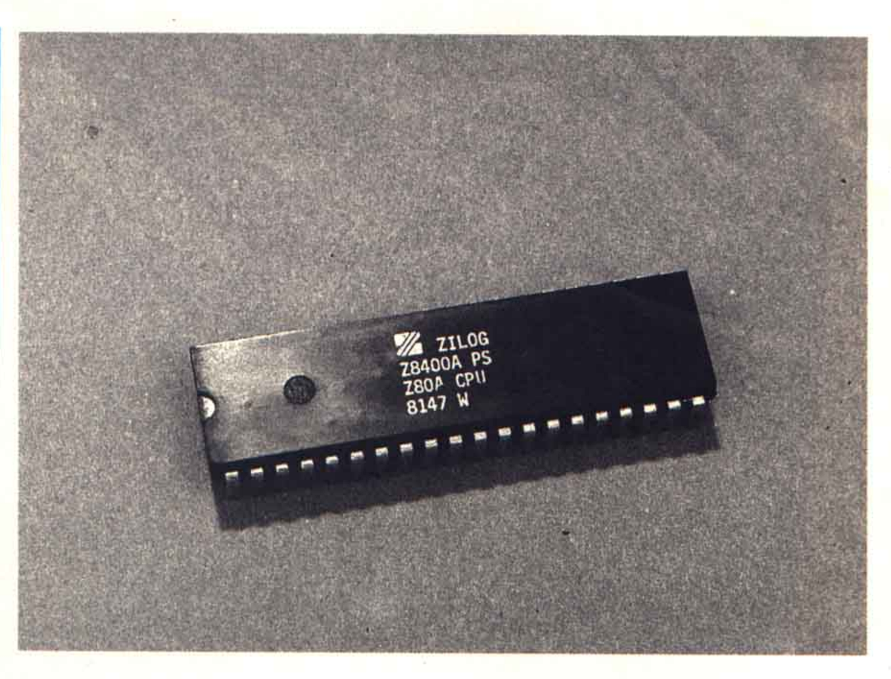
Como es el Z-80

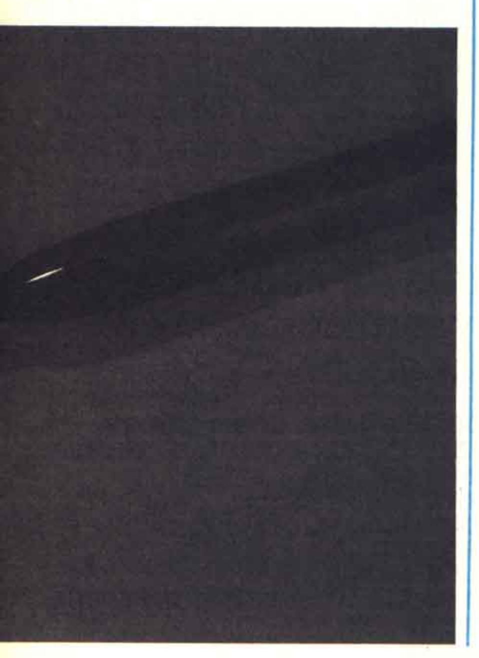
Por fin hemos llegado a la **MPU** más popular, el **Z-80**. Incorpora además de los registros del **8085**, un duplicado de todos ellos, incluido acumulador A y registro de *flags* F, excluyendo al contador de programas, que es único e igualmente el puntero de pila o *stack*. Este duplicado de los 8 registros A, F, B, C, D, E, H y L, no pueden usarse simultánemanete con los registros originales, sino sólo como sustitutos de ellos. El cambio se realiza mediante la instrucción EX. Esto permite un cambio de contexto muy rápido, sin necesidad de pasar por el *stack* o pila. Además, dispone de dos registros adicionales de 16 *bits*, índice X e índice Y y otros dos de 8 *bits*; R, de refresco para memorias dinámicas y el registro I que refleja el vector de interruptor del dispositivo que la ha provocado. El mecanismo de interrupción es diferente al **8085**. La interrupción vectorizada la posee también el **8085**, provocando un salto a la dirección de memoria que indica el periférico que efectúa la interrupción. En el **Z-80**, este vector queda recogido en el registro a tal efecto, siendo el programa el que actúa en consecuencia. Además, permite la programación de



tres modos de interupción, este último, de periféricos **Z-80**, otro compatible con el mecanismo de interrupción vectorizada del **8085** y un tercero para periféricos que no sean de la familia **Z-80**. Dispone además de las líneas para implementar un dispositivo de prioridades de interrupciones del tipo *"daisy-chain"*. El juego de instrucciones es claramente más potente que el del **8085**, con 78 instrucciones frente a 158 instrucciones del primero. Tienen instrucciones tan potentes que permiten mover y comparar bloques de datos con una única instrucción, que junto con las operaciones de lectura/escritura de periféricos, rotaciones de todo tipo y los modos de direccionamien-

La carrera tecnológica continúa, el próximo paso serán procesadores de 16/32 bits y simplicidad interior de los ordenadores personales, debido a la integración de diversos chips en uno solo.





to de las mismas hacen de este juego uno de los más potentes en MPÚs de 8 bits. La velocidad máxima del **Z-80** es de 6 Mhz, corriendo por tanto a 3 Mhz. La única carencia del **Z-80** es la de no disponer de instrucción de multiplicación entre su repertorio. De cualquier manera el **Z-80** es el más potente de los micros tradicionales de 8 bits, seguido por el **6809** muy de cerca. Establecer criterios comparativos con los demás resulta difícil, pudiéndo situar a la **6502** y al **6800** caso al mismo nivel. Entre estas dos categarías encontramos al **8085** y aunque menos conocida, se podría situar a la **65C02** en la misma posición, algo más avanzada. Esta visión rápida y sin pormenores de las MPÚs actuales más comunes se completa con un análisis en detalle del Z-80 y la comparación con su sucesor, el **Z-800**, debiendo introducir conceptos nuevos como el de memoria CACHE, lo que se deja para otra ocasión.

Francisco Cobian Schroeder



¡COMPRO, VENDO, CAMBIO...
Vendo Spectravideo SV-328 con cassette por 50.000 Ptas. Llamar al teléfono 215 84 30 de Barcelona y preguntar por Antonio Villegas, o escribir a C/ Aragón, 235. 08007 (Barcelona).
Club de usuarios de MSX. Para contactar y unirse a nosotros dirigirse a Isabel Linares. Los Chopos, 34, Las Gabias (Granada).
VENDO ordenador personal Spectravideo SV-328, junto con monitor de fósforo verde ycassetteSpectravideo, todo está nuevo. Llamar al teléfono (983) 33 80 69. Preguntar por Martín. (Valladolid)
Interesa contactar con usuarios MSX para intercambio de programas y listados. Además un cartucho del juego Step Up. Escribir a José Miguel Rodriguez. C/Rio Ser, 4 3-B. 17003 (Gerona).
Club de usuarios de MSX. Información y asesoramiento técnico con contactos en el extranjero. Escribir al Club de usuarios de MSX. Pza. Navío, 1. 28042 (Madrid).
RS



PUBLICACIONES ACERCA DEL MSX

Me dirijo a ustedes para decirles que ya era hora de que alguien se acordara de nosotros, los usuarios de ordenadores MSX.

Debido a que considero que el manual de instrucciones del ordenador que poseo deja bastante que desear, desearía preguntarles si existe en el mercado alguna publicación o algún libro que hable sobre el sistema MSX, sobre su programación, sus sentencias, etcétera. En resumen, un libro que explique detalladamente su programación.

Carmen Mejías
Sabadell

El principal motivo de MSX Magazine, fue la de ayudar e informar a los usuarios de este sistema, sobre las grandes posibilidades que ofrece este estándar. No pretendemos sustituir los manuales que acompañan cada ordenador, ni tampoco libros específicos.
Sin embargo, hasta el momento, la bibliografía existente sobre MSX está en inglés y pocos o ninguno en castellano. Por el momento, tomaremos buena nota del tema y procuraremos realizar todos los esfuerzos posibles para iniciar una sección dedicada a comentar libros de este sistema.

TRATAMIENTO DE TEXTOS PARA EL MSX

Recientemente he adquirido un ordenador **Philips** VG-8020 y la verdad es que me encuentro un tanto desorientado.

Por ese motivo adquirí la revista, donde especialmente me atrayeron dos secciones: BASIC MSX y la del



Software Comentado. En esta última, se hizo una descripción concisa y somera de una Base de Datos. Yo deseo saber si en algún número anterior han publicado algo sobre Tratamientos de Textos.

José María Ferreira
Málaga

Nos alegramos de que esas secciones le hayan interesado, pero todavía estamos empezando. El número que apareció en el mes de mayo fue el primero, por lo que, obviamente, no hemos realizado comentario alguno sobre Tratamientos de Textos. Estamos probando un par de ellos para, en futuros números, comentarlos a fondo. Uno es Tratamiento de Textos de la firma **AACKOSOFT,** que es holandesa y aparecerá en breve en nuestro país, y el otro programa que estamos probando es el **COMPOR,** de la casa **Philips,** este último aún no lo hemos visto a fondo.

DIRECTOR:
Juan Arencibia
COORDINADOR EDITORIAL:
Emiliano Juárez
REDACCION:
Fernando García, Santiago Gala, Ricardo García, Teresa Aranda, Francisco Mancera.
DISEÑO:
Ricardo Segura y Benito Gil.
Editada por
PUBLINFORMATICA S.A.
PRESIDENTE:
Fernando Bolín.
DIRECTOR EDITORIAL:
Norberto Gallego.
Administración:
PUBLINFORMATICA
GERENTE DE CIRCULACION Y VENTAS:
Luis Carrero.
PRODUCCION:
Miguel Onieva.
DIRECTOR DE MARKETING:
Antonio Gonzalez.
SERVICIO AL CLIENTE:
Julia Gonzalez.
Tel. 733 79 69
ADMINISTRACION:
Miguel Atance,
JEFE DE PUBLICIDAD:
María José Martín.
DIRECCION Y REDACCION:
C/ Bravo Murillo, 377 - 5° A
Tel. 733 74 13
28020 MADRID
PUBLICIDAD Y ADMINISTRACCION:
C/ Bravo Murillo 377 - 3° E
Tel. 733 96 62-96
Publicidad en Madrid:
Fernando Hernando
Publicidad en Barcelona:
Olga Martorell
C/ Pelayo, 12
Tel. (93) 301 47 00 Ext. 27-28.
08001 BARCELONA
Deposito Legal: M. 16.755-1985
Impreso en Héroes S.A.
C/ Torrelara, 8. 28016-MADRID.
Distribuye:
S.G.E.L. Avda. Valdelaparra s/n.
Alcobendas (Madrid).

SUSCRIPCIONES:
Rogamos dirija toda la correspondencia relacionada con suscripciones a:
MSX
EDISA: Tel, 415 97 12
C/López de Hoyos, 141-5.°
28002 MADRID
(Para todos los pagos reseñar solamente MSX)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
MSX
C/Bravo Murillo, 377-5.° A
Tel. 733 74 13 28020 MADRID

Si deseas colaborar en MSX remite tus artículos o programas a Bravo Murillo 377, 5.° A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

La versión española de Popular Computing

ORDENADOR POPULAR

LA REVISTA QUE INTERESA TANTO AL AFICIONADO COMO AL PROFESIONAL





Una publicación que informa con amenidad acerca de las novedades en el campo de las computadoras personales.

ORDENADOR POPULAR, la revista para el aficionado a la informática.

Ya está a la venta

Cómprela en su kiosco habitual o solicítela a:

ORDENADOR POPULAR

Bravo Murillo, 377
Tel. 7339662
28020 - *MADRID*



GoldStar MSX

MEMORIA RAM DE USUARIO: Una potente memoria de 64K le dará la fuerza necesaria para ejecutar los mejores programas del mercado.

CONECTORES DE EXPANSION: Aseguran la conexión a gran cantidad de periféricos como impresoras, diskettes y joysticks.

ROM y VIDEO ROM: Permiten al Goldstar ejecutar y trabajar con potentes programas de gráficos sin tener que utilizar la memoria RAM.

En el PORT DE CARTUCHOS podrá conectar todos los programas MSX existentes, simplemente introduciendo el cartucho —¡olvídese de esas complicadas cintas!

La FUENTE DE ALIMENTACION está incorporada al ordenador, de manera que no tendrá que manejar ni ocultar transformador alguno.

EL TECLADO es del tipo QWERTY, con la incorporación de teclas de función y del control del cursor.

EL SONIDO es una de las mejores características del Goldstar —con 5 octavas y un sin fin de tonos increíbles.

¡¡49.500 pts.!!





PAMPLONA:
C/Alfonso el Batallador, 16 (trasera)
Tel. 27 64 04 C. Postal 3107

SAN SEBASTIAN:
Plaza de Bilbao, 1.
Tel. 42 62 37 - Télex 38095-IAR
C. Postal 20005